# MSX linguaggio macchina

C.I. BURKINSHAW
R. GOODLEY

tecniche nuove

C.I. BURKINSHAW - R. GOODLEY

# MSX
# linguaggio macchina

C.I. BURKINSHAW - R. GOODLEY

# MSX
# linguaggio macchina

tecniche nuove

# PREMESSA

Il processore per la periferica video (VDP) originariamente compreso nelle specifiche MSX era il Texas Instruments 9929A. Questa unità è stata sostituita dalla più avanzata 9129. Ai fini della programmazione, però, le due unità sono da considerarsi identiche.

Il VDP costruisce l'immagine video a partire da numerose tabelle di dati localizzate nei 16K della RAM dedicata al video. Le locazioni delle tabelle in ciascuna delle modalità di visualizzazione vengono ritornate per mezzo della variabile di sistema BASE(X). Noi pensiamo che ciò sia una caratteristica comune a tutti i modelli con la Versione 1 del BASIC. Quindi le suddette locazioni della tabella, riportate nell'Appendice C, verranno ritenute valide negli esempi più significativi.

Il BASIC MSX consente ai programmi di utilizzare file in diversi e numerosi formati. Un'istruzione di cui non esiste documentazione, RUN "nome file", può forse essere utilizzata per caricare in memoria e far eseguire un programma memorizzato in formato ASCII (al posto di LOAD "nome file", R). Al momento di andare in stampa non siamo in grado di confermare lo stato di messa a punto di quest'istruzione e di conseguenza non verrà fatto alcun ulteriore riferimento ad essa.

# INDICE

Capitolo 1

# INTRODUZIONE

## Generalità. Organizzazione della memoria. L'interfaccia nastro. Le modalità di visualizzazione. Il processore video (V.D.P.) e l'integrato per la produzione di suoni.

### Generalità

Lo standard MSX è una specifica introdotta a metà del 1983 dalla Microsoft Inc.. Esso comprende il linguaggio BASIC, il sistema operativo e alcuni connettori esterni (che comprendono una cartuccia ROM e la porta per il joystick). Il software è intercambiabile tra tutte le macchine MSX (a patto che vi sia sufficiente memoria disponibile). Tutti i modelli (tranne lo Spectravideo SVI-728 e lo Yamaha CX5M) sono dotati di alimentazione interna. Sono altresì rigidamente specificate alcune caratteristiche non richieste nel sistema di base: una porta parallela del tipo "Centronics" per stampante, una seconda porta per il joystick, una interfaccia RS-232C e un secondo alloggiamento per cartucce di memoria.
Il sistema operativo (MSX-DOS) usa per i dischi il medesimo formato dell'MS-DOS, ma non vengono in alcun modo specificate le dimensioni dei dischi. L'MSX-DOS richiede un minimo di 64K di memoria RAM.
L'unità centrale di elaborazione è uno Z80-A, o equivalente, funzionante a 3.58 MHz, affiancata da un processore per la periferica video Texas Instruments TMS 9129 (per l'Europa). Il VDP ha 16K di RAM dedicata al video e consente di definire fino a 32 sprite, ciascuno di un solo colore. Tutti gli sprite appaiono davanti al piano di visualizzazione dei caratteri. Il VDP genera inoltre l'unico segnale d'interruzione del sistema. Questo avviene al termine di ogni scansione del video (approssimativamente 1/50 di secondo) e viene usato, per esempio, per l'analisi della tastiera.
Lo Z-80 ha una capacità di indirizzamento sufficiente per accedere ad uno spazio di memoria di 64K bytes. All'accensione, la ROM di sistema occupa i 32K di memoria ad indirizzo più basso. Viene inoltre fornito un minimo

aggiuntivo di 8K RAM di memoria utente. In pratica, il metodo di gestione della memoria comporta un'espansione di quest'ultima RAM aggiuntiva ad almeno 16K, ma quasi tutti i sistemi ne forniscono 32K o addirittura 64K.
L'interprete BASIC è in grado di utilizzare solo 32K di RAM - indipendentemente da eventuali RAM aggiuntive sulle schede - dei quali 28K circa disponibili per la memorizzazione dei programmi. Il BASIC MSX è il BASIC Standard della Microsoft (versione 4.5) con alcune estensioni, soprattutto per le elaborazioni grafiche e musicali.
Le principali caratteristiche comprendono un editor a schermo intero, comandi guidati da interruzioni e una precisione di 14 cifre sulle variabili del tipo di default.
Le capacità di produzione di suoni sono fornite da un generatore di suoni programmabile (PSG) General Instruments AY-3-8910, che consente l'utilizzo di tre voci su un'estensione di tre ottave. Il PSG inoltre gestisce l'input da joystick; la specifica minima per lo standard MSX indica una sola porta standard per joystick di tipo D.
Una PPI (Interfaccia periferica programmabile) 8255 Parallel I/O viene usata per selezionare i banchi di memoria che saranno visti dal processore in quattro "alloggiamenti", o "partizioni", di 16K (si veda oltre), e per la scansione della tastiera. Se viene fornita una porta per stampante (un'estensione standardizzata) essa è ancora servita dall'8255.
Una seconda estensione opzionale sottoposta a standard è un'interfaccia RS-232C. I componenti LSI necessari per quest'ultima sono il chip d'interfaccia di comunicazione 8251 e un temporizzatore ad intervallo programmabile 8253.
Al momento di andare in stampa i seguenti modelli sono stati annunciati per il mercato inglese:

Hitachi MB-H80
Toshiba HX-10
Sony HB-75 (48K ROM)
Sanyo MPC100
JVC HC-7GB
Yashica YC-64
Sega-Yeno DPH-64
Goldstar FC-20
Canon V-20
Yamaha CX5M (sintetizzatore + computer MSX)
Mitsubishi MLF-48
Mitsubishi MLF-80
Panasonic CF2800

## Organizzazione della memoria

Lo Z-80 può leggere e scrivere dati da e in un'area di memoria fino a sessantaquattro kilobyte. Il progetto MSX consente si scegliere questi 64K byte in una memoria più ampia. Questo banco di memoria è composta da quattro "partizioni" principali, o slot, ciascuna delle quali può contenere fino a 64K. La selezione della memoria viene gestita in blocchi, o pagine, di 16K ciascuno. L'area di memoria indirizzabile dallo Z-80 consiste quindi in quattro di queste pagine da 16K.
La porta A del PPI 8255 viene usata per determinare da quale delle suddette partizioni viene fornito ciascun blocco da 16K. I due bit meno significativi indicano quale tra esse fornisce il blocco da 0 a 16K e così via per i bit di ordine maggiore.
Si noti che un blocco può essere visto dal processore solamente alla stessa locazione che esso occupa nella partizione principale, vale a dire, per esempio, che il blocco della quarta partizione che copre gli indirizzi da 48K a 64K può apparire al processore solo a questi indirizzi e non da 0 a 16 o da 32 a 48K.
Le partizioni principali sono numerate da 0 a 3. Nella partizione 0 trovano posto i 32K della ROM del sistema operativo e dell'interprete. La maggior parte degli elaboratori MSX è dotata di 64K di RAM. Quest'ultima è di solito posta in un'unica partizione. Per esempio, il Sanyo MPC-100 usa a questo scopo il numero 3 mentre il Toshiba HX-10 usa la partizione numero 2. Una di quelle che restano è per la cartuccia nastro, mentre l'altra può essere configurata sia per un'altra cartuccia sia per una espansione del bus. La memorizzazione dei programmi BASIC inizia dall'indirizzo 32768.

## Porte di ingresso/uscita

Oltre a poter indirizzare 64K di memoria, lo Z-80 può effettuare delle operazioni di ingresso e uscita su 256 porte di otto bit ciascuna. Le porte dalla 128 alla 255 (&H80-&HFF) sono assegnate alle componenti del sistema e alle eventuali espansioni: RS-232C, stampante, processore del video, generatore di suoni, penna luminosa, ecc. Le restanti sono riservate. Il NMI (Interruzione non mascherabile) non è utilizzabile poiché il vettore locazione 66H viene usato nell'MSX-DOS.
Le operazioni di ingresso e uscita dovrebbero essere effettuate usando le procedure della ROM residente, poiché lo standard non garantisce che le locazioni di sistema siano le stesse sulle varie macchine. Un'eccezione può essere fatta per il processore video, dove può rivelarsi necessario un trasferimento rapido di dati.

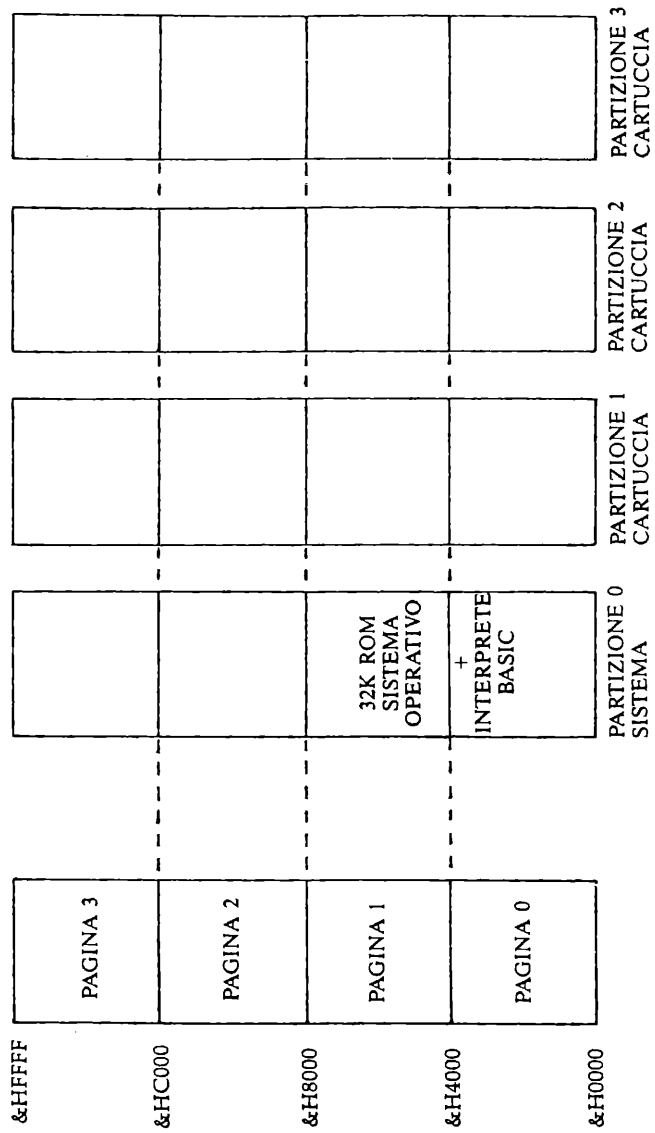



*Fig. 1.1 - Mappa della memoria.*

## Interfaccia nastro

Lo standard MSX non richiede la presenza di un'unità nastro dedicata. La velocità di trasferimento di default è di 1200 baud (circa 1200 bit al secondo) ma può essere elevata a 2400. La rilevazione e l'adeguamento ad una velocità differente da quella standard avviene automaticamente. La connessione esterna è una basetta DIN a 8 pin con la possibilità di controllo del motore del nastro. La modulazione avviene per mezzo di uno spostamento di frequenza sotto il controllo del software. I livelli ottimali per la registrazione e per la riproduzione vengono determinati dal registratore nastro; spesso però è necessario un livello assai vicino al massimo. Il motore dell'unità nastro può essere controllato per mezzo dell'istruzione MOTOR del BASIC.
Sono consentiti tre tipi di file:

1. Programmi memorizzati su cassetta. I comandi relativi sono CLOAD, CSAVE e CLOAD? per verificare.

2. File in formato ASCII, per i quali si usa SAVE e LOAD. Non è invece disponibile un comando per la verifica. I file di programma e i file ASCII possono essere mischiati.

3. Immagini del contenuto della memoria, per le quali si usa BSAVE e BLOAD. Anche in questo caso non è disponibile un comando di verifica.

## Modalità di visualizzazione

Il TMS 9129A è in grado di visualizzare un piano video con un massimo di 15 colori (più il trasparente) indipendentemente da qualsiasi sprite.
Sono disponibili quattro modalità di visualizzazione, due per i testi e due per la grafica:

1) Modalità testo con 40 * 24 caratteri, due colori e nessuno sprite, matrice di 6 * 8 punti per la visualizzazione del carattere, un insieme di 256 caratteri disponibili.

2) Modalità testo con 32 * 24 caratteri, 16 colori, matrice carattere di 8 * 8 punti, insieme di 256 caratteri disponibili.

3) Modalità grafica ad alta risoluzione con 32 * 24 caratteri e 16 colori. Si tratta essenzialmente di una modalità simile a quella per i testi, con 32 * 24 caratteri e con un insieme di caratteri su 768 celle per permettere uno schermo "bit-mapped" - più l'informazione extra del colore.

4) Modalità multicolore, 16 colori, ciascun blocco di 4 * 4 pixel può essere di diverso colore. Non sono disponibili i caratteri.

La modalità di visualizzazione viene selezionata usando l'istruzione BASIC 'SCREEN' oppure andando a modificare i tre bit che definiscono la modalità in due degli otto registri a sola scrittura del VDP (M3, bit 6 del registro 0, M1 ed M2, bit 3 e 4 del registro 1). In tutte le modalità, tranne in quella per i testi a 40 colonne, il colore del bordo può essere definito indipendentemente.
Il VDP scandisce i 16K della memoria RAM dedicata al video. Questa RAM è svincolata dallo spazio di memoria dello Z-80 e può essere letta o scritta dalla CPU solo attraverso il VDP. Quest'ultimo non fornisce alcuna funzione per lo scroll dello schermo.
L'istruzione BASIC per pulire lo schermo (CLS) è in qualche modo inusuale in quanto funziona in tutte le modalità di visualizzazione.

## Il VDP e gli integrati per la produzione del suono

**Texas TMS 9129A VDP (Processore video)**
Si tratta di un integrato a 40 pin standard "dual in line" (D.I.L.I.P.) che usa 16K di RAM dinamica. Solo il VDP può accedere alla RAM dedicata al video usando un bus dati unidirezionale a 8 bit. Il VDP viene controllato usando tre linee: RAS, CAS e R/W che derivano dal bus indirizzi e dalle linee di controllo del processore. Il VDP è inoltre connesso al bus dati di sistema.
Per effettuare da BASIC un'operazione di lettura o di scrittura sulla RAM del video è necessario usare le istruzioni VPOKE e VPEEK oppure le appropriate procedure in ROM a livello di linguaggio Assembler.

## La struttura video del VDP

In tutti i modi di funzionamento il VDP costruisce l'immagine da visualizzare principalmente a partire da due tabelle nella RAM video:

1. La tabella generatrice dei modelli.
2. La tabella dei nomi dei modelli.

Nella tabella generatrice dei modelli sono contenute le definizioni di ciascun carattere visualizzabile (cioè la descrizione, in termini di posizione dei

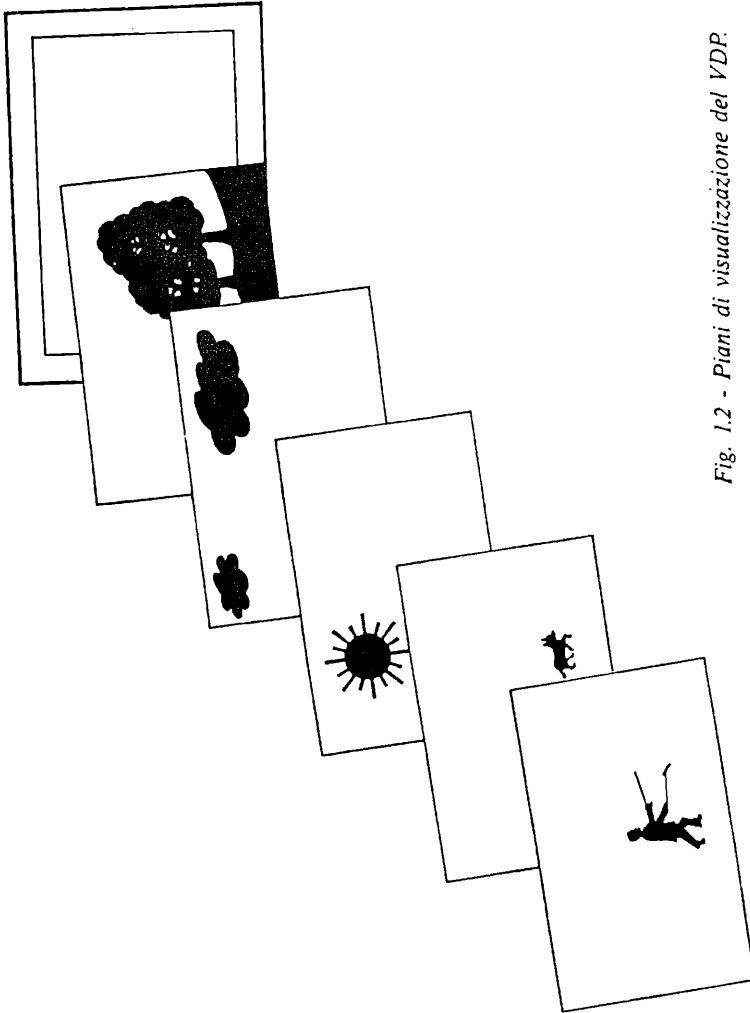



*Fig. 1.2 - Piani di visualizzazione del VDP.*

pixel, dei vari caratteri). L'indirizzo di partenza della tabella generatrice dei modelli in ognuna della modalità di visualizzazione è la seguente:

Modalità 0: TEXT 40 : 2048
Modalità 1: TEXT 32 : 0
Modalità 2: HRG : 0
Modalità 3: Multicolore : 0

In tutte le modalità, tranne quella multicolore, ciascun carattere è definito nella tabella generatrice dei modelli per mezzo di otto byte con cui viene specificata la geometria dei pixel - tutti i caratteri standard sono inoltre giustificati a sinistra con le due colonne più a destra e la riga di base lasciate vuote.
Questo viene illustrato nella figura 1.3
Nella modalità testo a 32 colonne possono essere usate fino a 256 definizioni di carattere, nel qual caso la tabella risulta lunga 2048 byte. La modalità HRG consente la definizione di 768 caratteri, producendo in tal modo una tabella lunga 6144 byte. La modalità testo a 40 colonne consente la definizione di 256 caratteri, ognuno di 8 byte, ma non vengono visualizzate le due colonne più a destra di ciascun carattere. In questo modo è possibile ottenere una schermata di 40 colonne con della matrici di carattere di 8 * 6 pixel e non di 8 * 8 come nella modalità HRG o testo a 32 colonne.
Nella modalità multicolore la tabella generatrice dei modelli contiene 192 entrate, ciascuna di otto byte, formate da quattro coppie di due byte ciascuna. Ciascuna coppia non definisce il carattere bensì il colore delle celle 4 * 4 nella matrice complessiva di 8 * 8.
La struttura della tabella generatrice di modelli viene illustrata nella figura 1.4.
*Esempio*: Nella modalità testo a 32 colonne la tabella generatrice inizia all'indirizzo 0 della VRAM, quindi il carattere con codice 65 - una "A" - viene definito dalle locazioni 65 * 8 = 520 fino alla 527.
La tabella dei nomi dei modelli determina quale carattere compare in ogni posizione sullo schermo (nella modalità multicolore vengono determinati i colori).
Nella modalità testo a 32 colonne ci sono 32 righe di 24 colonne e quindi 32 * 24 = 768 posizioni. Ciascuna di queste può contenere uno dei 256 possibili caratteri e di conseguenza la tabella dei nomi è lunga 768 byte - ognuno dei quali specifica quale carattere verrà messo in corrispondenza con una posizione.
Nella modalità testo a 40 colonne la tavola dei nomi per lo schermo ha 40 * 24 = 960 byte. Analogamente, ciascuno di questi descrive il carattere

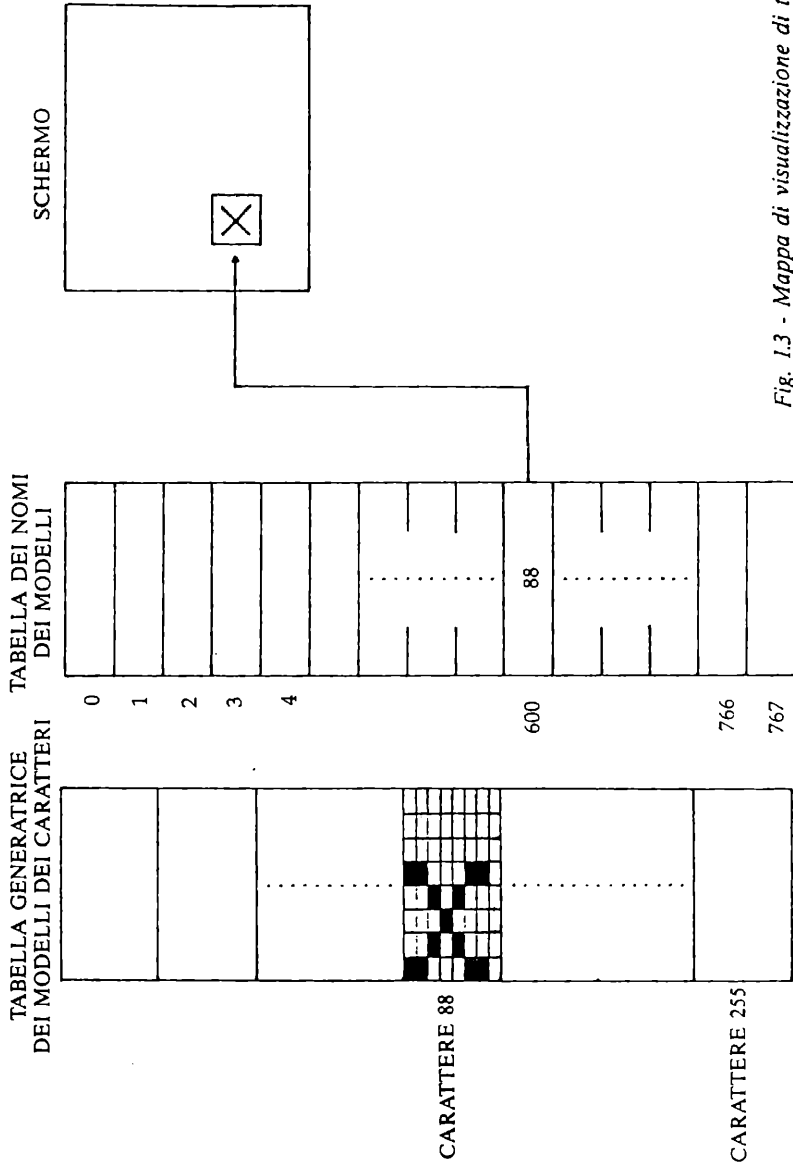



*Fig. 1.3 - Mappa di visualizzazione di testi.*

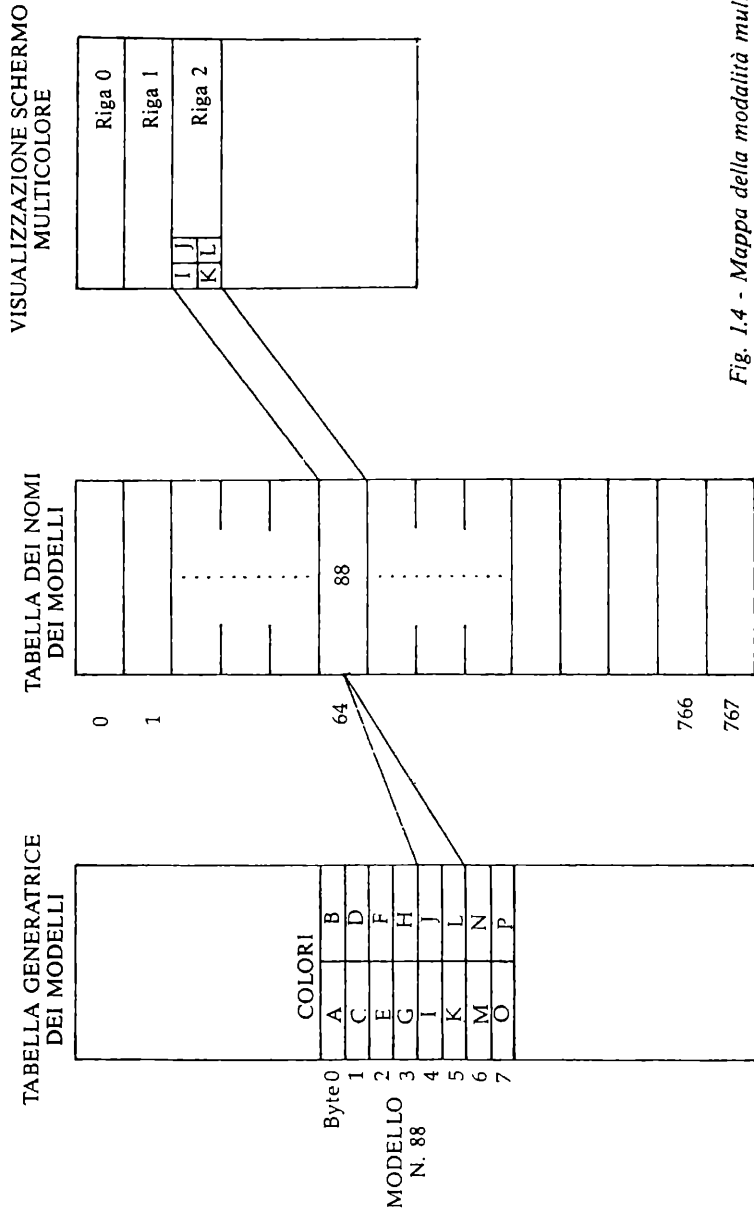



*Fig. 1.4 - Mappa della modalità multicolore.*

visualizzato in una determinata posizione dello schermo. Così come nella modalità testo a 32 colonne, la scelta è ristretta ai 256 modelli contenuti nella tabella generatrice dei modelli.

## Grafica ad alta risoluzione

Nella modalità HRG ci sono ancora una volta 768 posizioni nella tabella dei nomi, ma poichè in questa modalità sono disponibili 768 definizioni di carattere, come può un byte stabilire la selezione?
A tale scopo lo schermo è concettualmente diviso in tre zone e le prime 256 entrate nella tabella dei nomi selezionano i caratteri da 0 a 255, le seconde 256 selezionano i caratteri da 256 a 511 e le ultime 256 i caratteri da 512 a 768.
La tabella dei nomi nella modalità multicolore è lunga 768 bytes. Il valore contenuto in ciascun byte è un riferimento a un blocco di otto byte nella tabella generatrice dei modelli per il multicolore.
Due bytes di questo blocco indicano i colori dei blocchi 4 * 4 che formano la matrice del carattere. La colonna sullo schermo ove si trova la matrice carattere determina a quale dei quattro blocchi si fa riferimento. Se si trova sulla riga più alta vengono referenziati i primi due bytes, se si trova sulla seconda riga sono interessati i bytes 3 e 4 e così via, con la coppia successiva usata per la riga successiva. Quindi le righe 0, 4, 8, 12, 16, 20 faranno tutte riferimento ai primi due bytes.

### *Colore*

Benché il TMS 9129 VDP sia in grado di usare 15 colori più il trasparente (compresi il bianco e nero), nella modalità testo a 40 colonne possono essere usati solo due colori, e il bordo assume il colore dello sfondo.
In ambiente BASIC questi sono impostati (come per le modalità) per mezzo dell'istruzione COLOR, mentre in linguaggio macchina il registro a sola scrittura numero 7 del VDP determina entrambi i suddetti colori. Per modificare quest'ultimo è preferibile usare la procedura appropriata contenuta nella ROM.
Nella modalità testo a 32 colonne, la tabella generatrice dei modelli, con dimensioni 2K, è divisa in più insiemi di otto definizioni di carattere. Per ciascun insieme, il colore del primo piano e quello dello sfondo; sono determinati da un solo byte in una tabella generatrice di colori. Quindi la tabella dei colori è lunga 32 byte.

Il colore del primo piano è contenuto nel semi-byte più significativo e viceversa per il colore dello sfondo. Ne consegue che per visualizzare una lettera due volte con differenti colori, la definizione del carattere deve essere duplicata in un'altra posizione della tabella generatrice dei modelli: a quel punto ad essa può essere assegnato un diverso colore. Nella modalità HRG la combinazione sfondo/primo piano può essere specificata per ciascuna delle otto "linee" di ciascuna matrice carattere. A questo scopo sono necessari 768 * 8 = 6144 byte che vengono organizzati in un formato analogo a quello della tabella generatrice dei modelli. Questo metodo di allocazione dei colori rende problematica un'eventuale procedura per lo scroll del video.

*Sprite*

Uno sprite è un blocco costituito da un quadrato il cui lato è di 1, 2 o 4 celle carattere. È indipendente dal piano di visualizzazione e viene mosso facendo variare le coordinate X e Y che sono in relazione ad esso. Può essere acceso o spento come si desidera; inoltre ogni collisione tra due sprite viene rilevata automaticamente. Ogni sprite è dotato di una priorità e se una sezione non trasparente di uno sprite a più alta priorità passa davanti ad un altro, quest'ultimo viene parzialmente nascosto. Gli sprite consentono di ottenere con facilità effetti piuttosto sofisticati come, ad esempio, simulazioni tridimensionali.
Gli sprite non sono disponibili nella modalità testo a 40 colonne. In tutte le altre possono essere utilizzati sullo schermo fino a 32 sprite, con un massimo di quattro per ogni linea. Se si supera questo limite, solo i quattro sprite a più alta priorità vengono visualizzati sulla linea. Inoltre viene sollevata la condizione di "quinto sprite" ed il suo numero viene posto nel registro di stato del VDP.
Ciascuno sprite può essere di un unico colore e la sua priorità non può essere cambiata. Il BASIC mette a disposizione dell'utente due istruzioni per controllare gli sprite. La prima, SPRITE$(n) viene usata per definire un banco di modelli di sprite e la seconda, PUT SPRITE determina la posizione, il colore e il numero di ogni sprite visualizzato.
Questi comandi agiscono su due tabelle nella RAM video che contengono tutti i dati di visualizzazione degli sprite. La tabella dei modelli di sprite, che parte dalla locazione 14336 della VRAM (RAM Video) - indipendentemente dalla modalità di visualizzazione - contiene la definizione della forma per tutti gli sprite. La tabella degli attributi degli sprite contiene le coordinate X e Y, il colore e il modello di ogni sprite.
Gli sprite possono avere dimensioni sia di 8 * 8 pixel, sia di 16 * 16. Inoltre possono essere visualizzati con dimensione doppia di quella di definizione

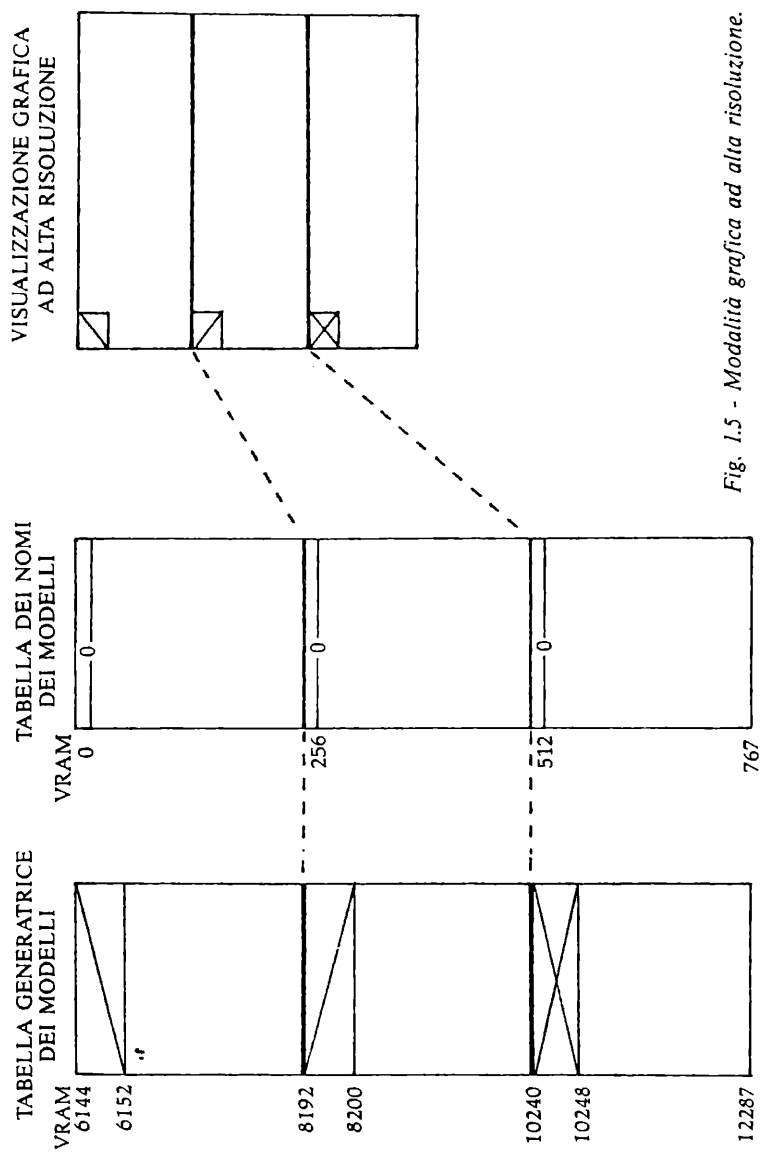



*Fig. 1.5 - Modalità grafica ad alta risoluzione.*

- quindi si arriva ad un massimo di 32 * 32 pixel. Il loro posizionamento è relativo all'angolo sinistro in alto dello schermo e possono essere mossi di un pixel alla volta.
La tabella dei modelli degli sprite può contenere fino a 256 blocchi di otto byte. Se gli sprite hanno dimensioni di 8 * 8 pixel (non si possono usare contemporaneamente quelli a 8 e a 16) allora possono essere definiti 256 modelli di sprite differenti - il modello 0 dalla locazione 14336 alla 14343, il numero 1 dalla 14344 alla 14351 e così via. Se invece si scelgono gli sprite a 16 pixel, ciascuno di essi viene definito usando quattro blocchi consecutivi di otto byte. Ciascun blocco definisce una sezione pari a un quarto dello sprite, nella sequenza:

1 3
2 4

Dopo la definizione della forma di uno sprite, esso viene posto sullo schermo tramite un'entrata della tabella degli attributi degli sprite. In tutti i modi di visualizzazione che consentono l'uso degli sprite, la tabella inizia alla medesima locazione della VRAM: 6912 e ha una lunghezza massima di 32 sprite * 4 byte = 128 byte. Ciascuna entrata serve ad indicare la posizione, il colore e il numero di modello per un singolo sprite. I primi due byte danno le coordinate Y e X mentre il terzo identifica la forma selezionando uno o più blocchi nella tabella dei modelli degli sprite. L'ultimo byte contiene nel semibyte meno significativo il colore di sfondo dello sprite. In aggiunta il bit più significativo consente di spostare a sinistra, quando è posto a 1, lo sprite di 32 pixel ("clock early bit").
Le collisioni tra sprite vengono gestite in BASIC usando le istruzioni SPRITE ON e ON SPRITE.
La figura 1.2 mostra la combinazione dei vari piani di visualizzazione prodotti dal VDP.

## General Instruments AY-3-8910 PSG (Generatore programmabile di suoni)

Si tratta di un generatore programmabile di suoni basato su registri che rende disponibili tre voci su un'estensione di otto ottave. Inoltre l'ingresso da joystick è gestito tramite due porte di I/O indipendenti, situate sulla scheda. Ogni voce ha un controllo separato di volume e può produrre un suono pulito oppure un rumore. Però la stessa frequenza di rumore e forma d'inviluppo viene usata per tutti i canali - se due canali producono rumore, essi producono lo STESSO rumore; può essere regolato solo il volume.

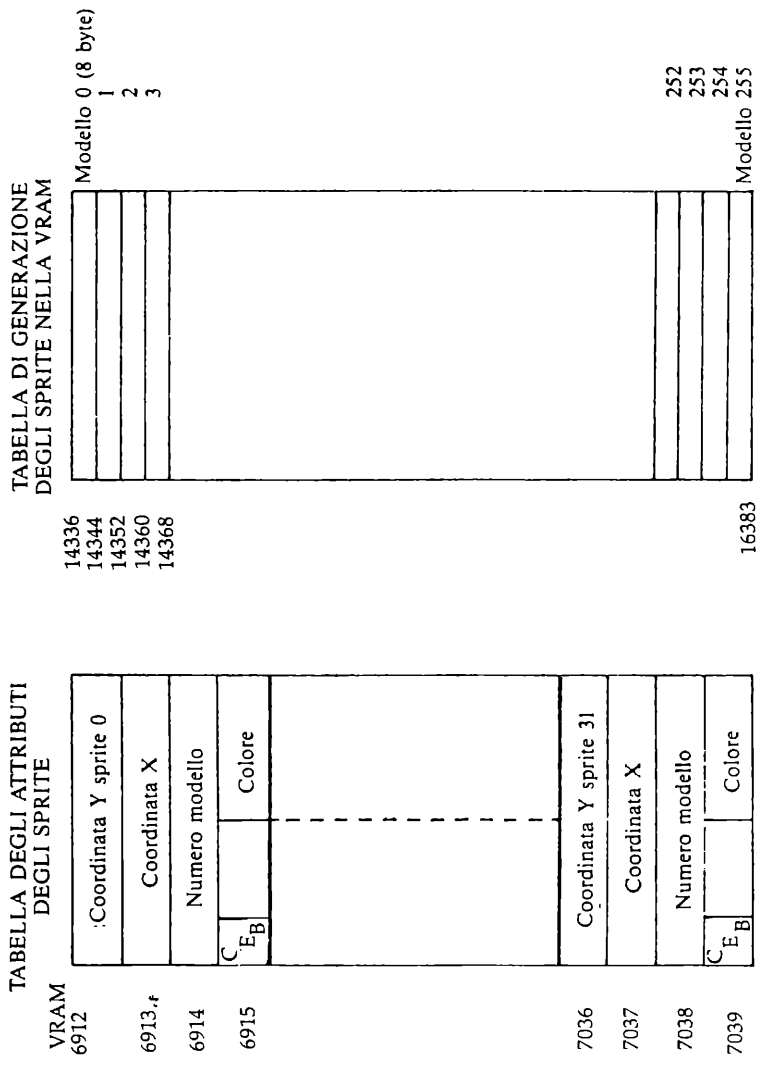



*Fig. 1.6 - Tabelle per gli sprite.*

Il PSG ha 16 registri interni di lettura/scrittura, due dei quali hanno funzioni di registri di memorizzazione dati per le due porte del joystick. I restanti registri possono essere divisi in 5 gruppi; in base alla loro funzione:

1. R0-R5: controllo della frequenza di canale.
2. R6: frequenza del generatore di rumore.
3. R7: registro di canale per la scelta tra rumore e suono pulito. I due bit più significativi determinano la direzione del trasferimento dati delle due porte di I/O.
4. R10-R12 (Ottale): registri per la scelta tra ampiezza del suono controllata dall'inviluppo e ampiezza del suono fissata.
5. R13-R15 (Ottale): registri per la selezione della forma dell'inviluppo e del periodo.

Le due porte di ingresso e uscita sono del tutto indipendenti dalle funzioni di generazione del suono del PGS.
Il BASIC MSX comprende un buon numero di istruzioni per "fare musica" per le quali molti parametri sono fissati per default. In combinazione con la capacità di generare interruzioni temporizzate, tutto quanto descritto permette di ottenere effetti musicali complessi con estrema facilità.

Capitolo 2

# IL BASIC MSX

## Descrizione. Variabili e funzioni. Comandi di grafica. Suoni. Memorizzazione dei programmi.

Il BASIC MSX è sostanzialmente il BASIC Microsoft standard versione 4.5 con numerose estensioni nell'area della grafica e della produzione di suoni. Le principali mancanze sono invece le Procedures, le strutture WHILE/WEND e REPEAT/UNTIL e la completa abbreviazione delle parole chiave.
Viene reso disponibile un editor a schermo intero che permette di correggere una qualsiasi linea dello schermo. La correzione effettiva della linea avviene quando viene battuto il carattere RETURN mentre il cursore si trova su un qualsiasi carattere della linea in questione. Nella fase di introduzione non viene effettuato alcun controllo di correttezza.
Una linea può essere composta al massimo da 255 caratteri e può contenere più di una istruzione.
Più istruzioni che si trovino sulla stessa linea devono essere separate dai due punti. Un'istruzione REM fa sì che l'esecuzione prosegua dalla linea successiva. I numeri di linea devono essere compresi nell'intervallo 0-65529, compresi gli estremi.
Gli spazi non sono necessari e vengono ignorati dall'interprete (tranne quando fanno parte di una stringa). Una conseguenza di questo fatto è che le parole chiave non possono essere parte di nomi di variabili (dei quali sono riconosciuti solo i primi due caratteri). Vengono considerate diverse le lettere maiuscole e quelle minuscole. La parole chiave possono essere scritte sia maiuscole che minuscole.
A fianco della tastiera principale vi sono due gruppi di tasti. Sulla destra il gruppo dei tasti per l'editing - i tasti con le frecce che indicano la direzione dello spostamento del cursore e le opzioni di inserimento e cancellazione. Sulla sinistra c'è un insieme di cinque tasti funzionali. Al momento dell'accensione questi tasti sono predefiniti con dieci funzioni. Quelle corri-

spondenti ai tasti funzionali "minuscoli" (cioè senza che sia premuto il tasto SHIFT) sono mostrate sull'ultima linea dello schermo. Questa visualizzazione può essere evitata, pur mantenendo la definizione delle funzioni, tramite il comando KEY OFF.
Per ridefinire uno qualsiasi dei tasti funzionali, si usa l'istruzione KEYx, "stringa". Per esempio, si può definire nel seguente modo il tasto funzionale 1 in modo che serva per mandare in esecuzione un programma:

```
KEY1,"RUN"+CHR$(13)
```

La visualizzazione di una lista di programma o un'esecuzione può essere interrotta premendo il tasto STOP una volta. Premendolo nuovamente viene riattivata l'azione precedentemente interrotta. Per terminare definitivamente l'operazione in corso si devono premere contemporaneamente lo STOP e il tasto CONTROL.
Per coloro che non hanno familiarità con il BASIC Microsoft è opportuno porre in evidenza una serie di caratteristiche:

1. La messa a punto dei programmi è semplificata dall'uso delle istruzioni TRON e TROFF. Se viene eseguito il comando TRON, direttamente o all'interno di un programma, questo fa sì che venga visualizzato il numero di ogni linea che viene eseguita; solitamente eventuali altre visualizzazioni sono sovrascritte.

2. Blocchi di linee possono essere cancellati con l'istruzione DELETE x, y, mentre le linee di un programma possono essere rinumerate tramite il comando RENUM x, y. Questa possibilità si rivela utile nell'identificare eventuali salti a numeri di linea non esistenti. In tal caso infatti il BASIC MSX non segnala un errore; viene invece eseguita la linea con numero immediatamente superiore a quello indicato nel salto.

3. Tutte le variabili che iniziano con un particolare carattere, o con un insieme di caratteri, possono essere dichiarate di un particolare tipo: DEFINT per variabili intere, DEFSNG per variabili in singola precisione, DEFDBL in doppia precisione e DEFSTR per le stringhe.

4. I vettori possono essere cancellati per mezzo del comando ERASE.

5. Al momento dell'accensione lo spazio allocato per la memorizzazione delle stringhe è di 200 byte. Un'ulteriore allocazione necessita dell'uso di un comando di CLEAR. Ad esempio, per riservare 1000 byte si usa CLEAR 1000.

6. La funzione FRE(0) ritorna l'ammontare della memoria disponibile per la memorizzazione del programma.

7. La funzione per le stringhe MID$ può essere usata per sostituire uno o più caratteri in un'espressione.

8. La struttura GOSUB / RETURN consente di specificare un indirizzo di ritorno: RETURN "numero di linea".

## Variabili e funzioni

I nomi delle variabili possono essere di qualsiasi lunghezza, ma devono iniziare con una lettera. Solo i primi due caratteri sono significativi. Le lettere minuscole vengono distinte da quelle maiuscole. Non è necessario inizializzare le variabili: viene assunto il valore zero. Le variabili di tipo vettore non devono necessariamente essere dimensionate, tranne quando il numero di elementi supera undici: quindi S(0) e S(10), per esempio, non hanno bisogno di un esplicito dimensionamento.
Una variabile che non è dichiarata di un tipo particolare si assume che sia reale in doppia precisione, e viene memorizzata con quattordici cifre significative. Le variabili dichiarate in singola precisione sono memorizzate con una massima precisione di sei cifre. Gli interi sono compresi tra —32768 e 32767.
Una variabile in singola precisione viene distinta tramite un punto esclamativo alla fine, come SP!, mentre un'intera viene contraddistinta da un segno di percentuale (%). Se è necessario dichiarare una variabile in doppia precisione, viene usato il simbolo #.
Come accennato in precedenza, le varianti dell'istruzione DEF possono essere utilizzate per definire globalmente i tipi delle variabili. Se ad esempio viene usata l'istruzione DEFINT A, allora tutte le variabili che iniziano con A verranno trattate come variabili intere. Inoltre non verrà fatta alcuna distinzione tra 'A' e 'A%'.
Il dimensionamento delle variabili è molto semplice: si consideri a questo proposito l'istruzione DIM X(20,40),Y(80,160) che definisce due matrici rettangolari; la prima ha 20 righe e 40 colonne mentre la seconda ne ha 80 per 160. Esiste il limite di 255 sul numero delle dimensioni, ma il numero degli elementi è limitato solamente dalla disponibilità della memoria.
Poiché una qualsiasi variabile vettore che abbia meno di 11 elementi non deve essere esplicitamente dimensionata, ne consegue che, per un utilizzo ottimale dello spazio di memoria, è opportuno dimensionare quelle che hanno un numero di elementi minore di 11.
Il BASIC MSX comprende un'istruzione SWAP x,y che serve per scambiare il contenuto delle variabili x e y. Le variabili in questione devono però essere del medesimo tipo. Un'altra istruzione in relazione con le variabili

è la VARPTR x che ritorna la locazione di memoria dell'oggetto specificato - se quest'ultimo è stato dichiarato.
Le variabili di sistema dell'MSX sono:

1. **TIME:** è, in un certo senso, l'orologio del sistema. Viene incrementato ogni 1/50 di secondo.

2. **BASE(n):** ritorna la locazione della tabella specificata della RAM video, indipendentemente dalla modalità di visualizzazione.

3. **VDP(n):** ritorna il valore del registro a sola scrittura specificato del VDP, o viceversa il registro di stato di quest'ultimo.

4. **SPRITE$(pattern #):** questa variabile viene usata per definire ognuna delle 256 possibili 8 * 8 forme di sprite, oppure una delle 64 forme 16 * 16. Contiene una stringa di 32 caratteri. Il codice di ciascuno di questi caratteri determina la configurazione dei bit in un byte della definizione dello sprite.

## Funzioni

Sono disponibili tutte le funzioni che siamo abituati a trovare nel BASIC. In particolare, se XYZ è il nome di una variabile arbitraria:

ASC(XYZ$), CHR$(XYZ), LEN(XYZ$), INT(XYZ), EXP(XYZ), ABS(XYZ), LOG(XYZ), HEX$(XYZ), OCT$(XYZ), BIN$(XYZ), SGN(XYZ), RND(XYZ), MID$(XYZ$,N,n), RIGHT$(XYZ$,n), LEFT$(XYZ$,n), STR$(XYZ), VAL(XYZ$), STRING$(n,XYZ$), INSTR(XYZ$,xyz$), ATN(XYZ), COS(XYZ), SIN(XYZ), SQR(XYZ), TAN(X), TAB(n), SPC(n), INKEY$, INPUT$(n), PEEK(XYZ), POKEXYZ, FRE(0), FRE" ", USRXYZ(n), VARPTR(XYZ), VARPTR(#XYZ), CINT(XYZ), POS(n) e LPOS(n).

Le estensioni comprendono:

1. CDBL(XYZ) e CSGN(XYZ) che convertono XYZ rispettivamente a doppia e singola precisione.

2. VPEEK(n) e VPOKEn,x sono le istruzioni equivalenti, per la RAM video, di PEEK e POKE.

3. STICK(n) e STRIG(n) ritornano la direzione o lo stato di trigger del joystick (oppure del cursore e della barra di spaziatura).

4. POINT(x,y) ritorna il colore del pixel che si trova alle coordinate x e y.

5. PLAY(canale) indica lo stato di una o di tutte le code musicali.

6. EOF(file#) ritorna —1 se è stata raggiunta la fine di un file sequenziale, altrimenti 0.

7. PAD(n): in questo caso n determina quale paramentro dello stato di un paddle a controllo tattile viene ritornato.

8. PDL(paddle#) ritorna il valore di un paddle.

## Istruzioni per la grafica

L'intera gamma dei 15 colori più il trasparente è disponibile in tutte le modalità di visualizzazione (si veda l'Appendice B). Le istruzioni per la grafica possono essere divise in tre categorie:

1. Formattazione generale e colore.
2. Sprite.
3. Alta risoluzione.

## Comandi di uso generale

La modalità di visualizzazione sullo schermo viene impostata con l'istruzione SCREEN:

SCREEN0 modalità testo 40 * 24
SCREEN1 modalità testo 32 * 24
SCREEN2 modalità grafica ad alta risoluzione (HRG)
SCREEN3 modalità multicolore

Questi comandi possono essere utilizzati anche per impostare la dimensione degli sprite, la velocità di trasferimento dati su cassetta e le opzioni per la stampante. Quando vengono usate per selezionare una modalità testo, l'insieme dei caratteri viene copiato dalla ROM alla VRAM. L'istruzione SCREEN non deve essere usata qualora l'utente abbia la necessità di fare riferimento ad un qualsiasi carattere ridefinito.

## Modalità testo

La larghezza effettiva dello schermo sia per la modalità a 40 colonne sia per quella a 32, può essere diminuita con l'istruzione WIDTH (ad esempio WIDTH20 per avere lo schermo su 20 colonne soltanto). Si noti che in entrambe le modalità testo la colonna più a destra e quella più a sinistra non vengono usate. La maggior parte dei ricevitori TV non visualizzano la colonna più a sinistra.
In entrambe le modalità i caratteri possono essere messi in una colonna

particolare scrivendo sulla tabella dei nomi della RAM video. Ad esempio, per mettere una A maiuscola nella prima riga della due colonne più a sinistra nella modalità a 40 colonne, si usa la VPOKE delle locazioni 0 e 1 con il valore 65.
I colori globali per una qualsiasi modalità di visualizzazione vengono impostati tramite l'istruzione COLOR. Il formato della sua sintassi è il seguente:

COLOR primo-piano, sfondo, bordo

L'impostazione di default è 15, 4, 4. Nel modo a 40 colonne il colore del bordo non può essere specificato separatamente e assume quello dello sfondo. Nei modi schermo 1, 2 e 3 si possono ottenere uno sfondo ed un bordo color grigio inchiostro con l'istruzione COLOR14,1,1.
Si noti che nelle modalità grafiche il nuovo colore dello sfondo diventa effettivo solo dopo un'istruzione CLS.
Nella modalità testo a 40 colonne, il VDP non consente di usare altro colore. Non è possibile usare gli sprite. Viceversa nel modo a 32 colonne il VDP visualizza il testo in un colore differente da quelli scelti per mezzo dell'istruzione COLOR. Però a questo scopo non esiste comando BASIC. Quindi per ottenere sullo schermo più di un colore di primo piano o di sfondo; è necessario apportare dei cambiamenti alla tabella dei colori della VRAM. Per far ciò si usa VPOKE. Un'operazione del tutto simile deve essere effettuata per ridefinire dei caratteri.
In entrambe le modalità testo, la posizione del cursore viene determinata tramite l'istruzione LOCATE X,Y. Nel modo a 40 colonne è possibile visualizzare fino a 37 caratteri su una linea. Nonostante siano accettate posizioni di colonna fino a 255, solo quelle comprese nell'intervallo 0-36 hanno effetto. Per la modalità a 32 colonne l'intervallo accettabile è da 0 a 28. Un'alternativa all'istruzione LOCATE0,0 è la stampa su schermo di CHR$(11). Il comando LOCATE controlla inoltre la visualizzazione del cursore:

LOCATE2,4,0 fa sì che esso non venga visualizzato

LOCATE2,4,1 abilita la visualizzazione del cursore

Il comando PRINT, che può essere abbreviato con "?", usa i caratteri standard per la formattazione:

1. Il punto e virgola implica che il carattere di "ritorno carrello" (CR o carriage return, CHR$(13)) non venga stampato dopo l'ultima entità da visualizzare. In questo modo la prossima visualizzazione non partirà dalla prima colonna della linea successiva.

2. Una virgola implica che la prossima entità visualizzata sia allineata con il prossimo carattere (o zona) di tabulazione presente sulla linea. Ogni zona di tabulazione è ampia 14 colonne.

3. Il segno "+" serve per concatenare delle stringhe. Per esempio:

A$=B$ + "XYZ"
TAB(n) e SPC(n) devono seguire l'istruzione di PRINT cioè
PRINTSPC(4);"NOME
Si noti che il delimitatore finale " può essere assente.

La variante PRINT USING consente di stampare tabelle numeriche o alfanumeriche in un dato formato. I quattro principali caratteri di controllo sono:

1. Il punto esclamativo (!): implica che venga stampato solo il primo carattere di una stringa, come:

PRINT USING "!"; "NOME" darebbe come risultato N

2. La barra (\): viene usata nella forma "\\" per specificare il numero dei caratteri della stringa che devono essere stampati. Il numero dei caratteri stampati è infatti due più il numero di spazi tra le due barre, come, ad esempio:

PRINT USING "\\"; "NOME" darebbe come risultato NOM

Se la lunghezza del campo supera quella della stringa, vengono aggiunti degli spazi.

3. Il segno di 'e' commerciale (&): viene usato per inserire una seconda stringa in una prima a partire dalla posizione in cui è posta la &; quindi:

Q$="NOME":?USING"IN &";Q$ darebbe come risultato IN NOME

Si noti che nel caso venissero fornite più sottostringhe, la sequenza mostrata in precedenza verrebbe ripetuta per ciascuna di esse. Per esempio:

Q$="A":Z$="B":?using"SU &";Q$,Z$ darebbe come risultato SU ASU B

4. Il segno di diesis (#): consente di stampare i numeri fissando il numero di cifre prima e dopo la virgola decimale. Se il valore è più corto, vengono aggiunti degli 0. Per esempio:

?USING"# # #.# #";2,4.684 darebbe 2.004.68

Se viceversa il campo è troppo corto per contenere il dato, allora viene stampato il segno % davanti al valore, oppure viene arrotondato. Per esempio:

?USING" # #";224444 darebbe come risultato %224444
?USING" # #. # #";22.1234 darebbe 22.12

Se il campo è troppo grande, il valore viene giustificato a destra o vengono aggiunti degli zeri. Per esempio:

?USING" # #. # #";22.2 darebbe come risultato 22.20
?USING" # # # #";22 darebbe  22

Il numero non può essere più lungo di 24 cifre, nel qual caso verrebbe segnalata una situazione di errore.
Altri caratteri di controllo per i numeri, usati insieme al segno " # ", sono: **, ££, la virgola, +, — e ^^^^ (detto "carat").
Un segno più (+) alla destra o alla sinistra del simbolo diesis (#) implica che venga stampato il segno del valore numerico, rispettivamente alla destra o alla sinistra del valore stesso.
In modo analogo, un segno negativo dopo l'ultimo di una serie di " # " provoca la stampa del segno meno dopo il valore, se questo è negativo. Quindi:

?USING" # #—";—2 stamperà 2—

La combinazione doppio asterisco (**) viene usata sulla sinistra dei simboli # e fa sì che gli spazi iniziali, che risultano da una giustificazione a destra di un valore, vengano riempiti con asterischi; quindi:

?USING"** #. #";4.2 dà come risultato **4.2
?USING"** # #. #";4.2 dà invece ***4.2

Il segno di doppia lira (che qui indichiamo con ££) provoca la stampa del simbolo di lira prima del valore numerico in questione. Questo carattere di controllo non può essere usato in coincidenza del "carat" (^^^^).
Se si pone una virgola alla sinistra del punto decimale, il numero viene stampato con una virgola a sinistra di ogni gruppo di tre cifre, vale a dire:

?USING" # # # #,. #";2222.2 provoca la stampa di 2,222.2

Infine il quadruplo carattere (^) viene usato per stampare un valore nume-

rico in forma esponenziale; come esempio si consideri:

?USING "# #. #^^^^";22.4 implica la stampa di 2.2E+01

Lo schermo può essere pulito in tutte le modalità di visualizzazione mediante il comando CLS. In entrambe le modalità testo, un'alternativa è la stampa di CHR$(12). La coordinata Y del cursore viene ritornata dall'istruzione CSRLIN e la coordinata X da POS(x) (dove x è un argomento fittizio, che può quindi essere sostituito da un qualsiasi valore).

## Manipolazione della RAM del video

Per ottenere che i caratteri siano visualizzati con colori diversi da quelli impostati globalmente per mezzo del comando COLOR nella modalità testo a 32 colonne, si deve andare a modificare il byte appropriato nella tabella dei colori della VRAM, scegliendo una nuova combinazione primo piano/sfondo.
Come già accennato, ciascun dei 32 byte della tabella dei colori determina i colori di un insieme di 8 dei 256 caratteri disponibili. Come conseguenza di ciò, per visualizzare del testo in colori diversi da quelli di default è necessario copiare parte dell'insieme dei caratteri in un'altra sezione della tabella di generazione dei modelli.

Tabella generatrice dei modelli : 0 -2047

Tabella dei colori : 8192 - 8223

Tabella degli attributi degli sprite : 6912 - 7039

Tabella dei nomi : 6144 - 6911

Tabella dei modelli degli sprite : 14336 - 16383

*Tavola 2.1 - Allocazione della memoria nella modalità testo a 32 colonne*

Si noti che i caratteri grafici occupano i primi 2 blocchi di definizione nella tabella dei modelli. Questo perché i primi 32 codici carattere non sono stampabili. I caratteri grafici possono quindi essere visualizzati sullo schermo solo stampando (PRINT) CHR$(1) + CHR$(65). Un altro carattere non standard ha codice 255: provoca la visualizzazione inversa del carattere su cui è posizionato il cursore. Viene aggiornato circa ogni 1/50 di secondo.

**Esempio: testo multicolore**

Per ottenere che un testo venga visualizzato in lettere maiuscole con primo piano giallo e sfondo nero, sono necessari i seguenti passi:
Copiare le 26 definizioni dei caratteri in un'altra sezione della tabella di generazione dei modelli. L'insieme di caratteri viene definito usando 8 byte per ciascun carattere. I caratteri dal 65 al 90 seguono la sequenza indicata in Appendice B.
Nella modalità testo a 32 colonne la tabella dei modelli si estende, nella VRAM, dalla locazione 0 fino alla 2048. Poiché ogni carattere necessita di otto byte di definizione, il primo byte che deve essere copiato si trova alla locazione:

```
ASC("A") * 8 = 520
```

e il byte finale della definizione dei caratteri sarà ASC("Z")*8 + 7 = 727
I 'nuovi' caratteri maiuscoli sovrascriveranno parte dell'insieme di caratteri esistenti. Sostituiremo i caratteri 145-170, che sono definiti da 145 * 8 = 1160 a 170 * 8 + 7 = 1367, come segue:

```
10 SCREEN 1:FOR X=0 TO 207:VPOKE 1160+X,VPEEK(520+X):NEXT
```

I quattro byte della tabella dei colori che determinano i colori che assumeranno i 32 caratteri dal 144 al 176 risiedono alle locazioni 8210-3. Il valore contenuto dai quattro bit più significativi in ciascuno di questi determina il colore del primo piano, mentre quelli meno significativi influiscono sul colore dello sfondo.

| PRIMO PIANO | SFONDO | |
|---|---|---|
| | | = 10 * 16 + 1 =161 |
| Giallo: 10 | Nero: 1 | |

*Tavola 2.2 - Entrata della tabella del colore*

Quindi la prossima linea di programma è (si veda la tavola 2.2):

```
20 FOR X=0 TO 3:VPOKE X+8210,161:NEXT
```

Se ora si prova a far visualizzare il CHR$(145) apparirà una 'A' gialla in campo nero. Si noti che se si usa il comando COLOR tutti i 32 byte della tabella del colore sarebbero impostati con i colori prescelti. Per ottenere

nuovamente dei colori alternativi sarebbe necessario ripetere la linea 20. I caratteri sono giustificati a sinistra: quindi quando vengono visualizzati su uno sfondo che contrasta con il colore dello schermo, può succedere che la parte sinistra del carattere appaia confusa.
Questo metodo per determinare i colori dello schermo, sebbene macchinoso per certi aspetti, permette di realizzare delle schermate efficaci dal punto di vista grafico in modo piuttosto semplice. Per esempio, una schermata che consista principalmente di 4-8 caratteri può essere 'animata' o resa lampeggiante manipolando solo due byte della tabella del colore.
Il programma che viene presentato di seguito consente di ridefinire interamente l'insieme dei caratteri, di salvarlo e caricarlo da nastro. I tasti di controllo del cursore sono usati per muoversi sulla matrice dei caratteri e la barra di spaziatura viene utilizzata per invertire il valore del particolare bit su cui si è posizionati.

## ESEMPIO DI PROGRAMMA PER LA DEFINIZIONE DELL'INSIEME DEI CARATTERI

```
 10 '*****************************************************
 20 '****  PROGRAMMA DI UTILITÀ PER LA DEFINIZIONE    ****
 30 '****  DELL'INSIEME DEI CARATTERI CON FUNZIONI    ****
 40 '****  DI SALVATAGGIO, CARICAMENTO E COPIA        ****
 50 '*****************************************************
 60 '
 70 'USATE IL CURSORE PER MUOVERVI NELLA MATRICE1
    DI DEFINIZIONE.
 80 'PER CAMBIARE IL VALORE DEL BIT PREMETE LA BARRA
    DI SPAZIATURA.
 90 'NOTATE CHE IL COMANDO SCREEN OPERA UN RESET
    SULLA RAM VIDEO.
100 '
110 FOR X=0TO12:READA$,B$:POKE38000!+X,VAL("&H"+
    A$):POKE38200!+X,VAL("&H"+B$):NEXT
120 DEFUSR=38000!:DEFUSR2=38200!
130 DATA 21,21,00,40,00,9C,11,11,40,00,9C,00,01,01,00,00,08,08,
    CD,CD,59,5C,00,00,C9,C9
140 ON KEY GOSUB 240,270,290,430:KEY(1) ON:KEY(2) ON:KEY(3)
    ON:KEY(4) ON
150 ON STRIG GOSUB 400:STRIG(0) ON
160 KEY OFF:COLOR14,1,1:SCREEN 1,0:CLS
170 LOCATE2,1:PRINT"F1..SAVE F3..CHANGE CHR$":LOCATE
    2,3:PRINT "F2..LOAD F4..COPY CHR$"
```

```
180 FORX=0TO7:VPOKE14336+X,VPEEK(224+X):NEXT:PUT
    SPRITE 0,(144,47),8,0
190 FOR X=373TO375:VPOKEX,0:NEXT:VPOKE371,24:VPOKE
    372,24
200 A$=STRING$(8,46):FORX=0TO7:LOCATE16,X+6:PRINTA$:NEXT
210 GOSUB 290:X1=1:Y1=1
220 X2=STICK(0):IF X2=0 THEN 220
230 ON X2 GOTO 350,220,360,220,370,220,380
240 Z=USR(2):LOCATE2,20:PRINT"RECORD THEN RETURN"
250 A$=INPUT$(1):IF ASC(A$)<>13 THEN 250 ELSE BSAVE"CAS:",
    40000!,42047!
260 LOCATE 2,20:PRINTSTRING$(28,32):RETURN
270 LOCATE 2,20:PRINT"LOADING...":BLOAD"CAS:":Z=USR2(2)
280 LOCATE 2,20:PRINTSTRING$(28,32):RETURN
290 LOCATE 0,20:PRINT"ENTER # OF CHR$<RET>":GOSUB
    470:CN=XX
300 FOR X=0TO7:A$(X)=BIN$(VPEEK(CN*8+X)):NEXT
310 FORX=0TO7:IF LEN(A$(X))<8 THEN A$(X)=STRING$
    (8—LEN(A$(X)),79)+A$(X)
320 FOR BT=1TO8:LOCATE15+BT,X+6:IF MID$(A$(X),BT,1)="1"
    THEN PRINTCHR$(219) ELSE PRINTCHR$(46)
330 NEXT:NEXT:LOCATE2,11:PRINT"CHARACTER";CN:LOCATE7,13:
    PRINT CHR$(CN)
340 RETURN
350 IF Y1=1 THEN 220 ELSE Y1=Y1—1:GOTO390
360 IF X1=8 THEN 220 ELSE X1=X1+1:GOTO390
370 IF Y1=8 THEN 220 ELSE Y1=Y1+1:GOTO390
380 IF X1=1 THEN 220 ELSE X1=X1—1:GOTO390
390 PUT SPRITE 0,(136+X1*8,39+Y1*8),8,0:FORX=1TO80:NEXT:
    GOTO220
400 LOCATE 15+X1,5+Y1:IF MID$(A$(Y1—1),X1,1)="1"THEN
    MID$(A*(Y1—1),X1,1)="0":PRINTCHR$(46) ELSE MID$(A$
    (Y1—1,X1,1)="1":PRINTCHR$(219)
410 NN=CN*8+Y1—1:IF MID$(A$(Y1—1),X1,1)="1"THEN
    VPOKENN,VPEEK(NN)OR(2^(8—X1)) ELSE VPOKENN,VPEEK(NN)
    AND(255—(2^(8—X1)))
420 RETURN
430 LOCATE0,20:PRINT"CHR$ TO COPY<RET>":GOSUB470:C1=XX
440 LOCATE0,20:PRINT"CHR$ TO BE REPLACED":GOSUB470:C2=XX
450 FORX=0TO7:VPOKEC2*8+X,VPEEK(C1*8+X):NEXT
460 LOCATE0,20:PRINTSTRING$(28,32):RETURN
470 C1$=""
```

```
480 X$=INKEY$:IFX$="" THEN480
490 IF ASC(X$)<32 AND ASC(X$)>27 THEN 480
500 IF ASC(X$)< >13 THEN C1$=C1$+X$:GOTO480 ELSE
    XX=VAL(C1$)
510 IF XX>255 OR XX<0 THEN 470
520 LOCATE0,20:PRINTSTRING$(28,32):RETURN
```

## Sprite

Le istruzioni e le variabili fondamentali del BASIC per la manipolazione degli sprite sono:

1. SPRITE$(numero modello)=XYZ$
2. PUT SPRITE n,(X,Y),colore,numero modello
3. PUT SPRITE n,STEP(x,y),colore,numero modello
4. SCREEN modo,dimensione sprite
5. SPRITE ON/OFF/STOP
6. ON SPRITE GOSUB

La dimensione degli sprite può essere sia di 8 * 8 pixel sia di 16 * 16. Inoltre entrambe le suddette dimensioni possono essere raddoppiate. L'istruzione SCREEN determina la dimensione che verrà usata:

0=8*8
1=8*8 raddoppiato
2=16*16
3=16*16 raddoppiato

quindi, ad esempio, SCREEN 2,2 provoca l'attivazione della modalità ad alta risoluzione grafica (HRG) con gli sprite di dimensione 16 * 16 pixel. Uno sprite di 8 * 8 pixels è definito, come un carattere, per mezzo di otto byte. Uno di dimensioni 16 * 16 viene definito usando quattro blocchi di otto byte:

```
1     17
|     |
8     24

25    9
|     |
32    16
```

Questi ultimi sono definiti tramite l'istruzione SPRITE$. Ciascuna defi-

nizione di modello di sprite viene introdotta in forma di stringa di 8 o di 32 caratteri, dove ogni numero corrispondente al carattere rappresenta un byte del modello.

Per esempio, per definire uno sprite 8 * 8 che rappresenti il segno '—':

```
A$=CHR$(0):B$=CHR$(126)

SPRITE$(0)=A$+A$+A$+B$+B$+A$+A$+A$
```

Infatti la configurazione dei bit che rappresentano 126 in numerazione binaria è la seguente:

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 1 | |

In alternativa all'uso dell'istruzione SPRITE$ si potrebbe usare la VPOKE per scrivere i primi otto byte della tabella dei modelli per gli sprite:

```
FORX=14336 TO 14343:VPOKEX,0:NEXT:VPOKE14339,126:
VPOKE14340,126
```

Se sono attualmente in uso gli sprite 8 * 8 si possono definire fino a 256 modelli di sprite, mentre con quelli a 16 * 16 pixel ne sono consentiti solo 256/4=64.
La possibilità di definire sprite di dimensioni così piccole facilita l'uso di un "insieme di caratteri" basato sugli sprite. È infatti possibile ottenere testo di dimensioni doppie e a più colori usando il suddetto insieme di caratteri unitamente all'opzione di raddoppio delle dimensioni. Sebbene esista il limite di quattro lettere per ogni linea dello schermo, questo approccio evita i problemi associati con la giustificazione a sinistra dell'insieme di caratteri residente nella RAM di sistema.
Uno sprite può essere visualizzato sullo schermo e fatto muovere usando una delle due forme dell'istruzione PUT SPRITE. Lo schermo è indirizzabile secondo coordinate che valgono 0,0 nell'angolo superiore sinistro, 0,190 sull'angolo inferiore sinistro e 255,0 nell'angolo superiore destro.
Una qualsiasi coppia di coordinate si riferisce al pixel in alto a sinistra dello sprite. Per poter far apparire lentamente lo sprite stesso entrambe le coordinate assumono come estremo inferiore —32. In questo modo è possibile posizionare sprite "fuori" dello schermo.
In ogni caso valori delle coordinate fuori dello schermo, ma comprese tra —32768 e 32767 non provocano una situazione di errore. Infatti lo sprite verrà posizionato, in tal caso, secondo le coordinate che si ottengono dai

valori dati, dividendoli per 256 e considerando il resto della divisione intera. L'unica eccezione si ha quando la coordinata Y assume il valore 208. Infatti quando il VDP si trova in questa situazione provvede a togliere dallo schermo lo sprite in questione e tutti quelli di priorità inferiore. Si tratta di un metodo efficace per far lampeggiare gli sprite. Il formato standard delle istruzione PUT SPRITE è il seguente:

```
PUT SPRITE piano, (X,Y), colore, numero del modello
```

Il piano 0 è il piano a più alta priorità: ne consegue che ogni parte non trasparente di uno sprite che si trovi nel piano 0 si sovrapporrà a qualsiasi cosa vi sia sullo schermo nella medesima posizione. Il piano a priorità più bassa è il 31. Per ogni piano è consentito un solo sprite.
Si ha inoltre a disposizione la stessa gamma di colori che si usa nell'istruzione COLOR. Si noti che ogni sprite può essere di un solo colore. Se non lo si specifica, viene usato il colore corrente di primo piano. Se non vengono specificate le coordinate lo sprite viene posizionato alle coordinate 209,0. Un'eccezione a questo comportamento si ha nel caso in cui uno sprite sia stato posizionato in precedenza sul piano in questione: in questo caso si usano le precedenti coordinate.
Ad esempio: PUT SPRITE0,(40,40),10,0:PUT SPRITE0,,10,2 provoca la sostituzione sul piano 0, in posizione 40,40, dello sprite modello 0 con quello modello 2.
L'istruzione PUT SPRITE può assumere anche il seguente formato:

```
PUT SPRITE Piano, STEP(x,y), colore, numero del modello
```

che consente di variare le coordinate dello sprite in relazione a quelle correnti. Gli sprite hanno priorità maggiore rispetto al piano di visualizzazione dei caratteri: ogni parte non trasparente di uno sprite si sovrapporrà quindi alla schermata sottostante. Così come la limitazione di un solo colore per ciascuno sprite, anche questa priorità è intrinseca nel VDP.
L'istruzione PUT SPRITE aggiorna un'entrata di quattro byte nella tabella degli attributi degli sprite, nella VRAM. In tutte le modalità che consentono l'uso degli sprite, la suddetta tabella inizia all'indirizzo 6912 ed è lunga 32 * 4 = 128 byte. I quattro byte di ciascuna entrata sono così allocati:

Byte 1 Coordinata Y
  2 Coordinata X
  3 Numero del modello
  4 Colore/"Clock early bit"

In BASIC, la coordinata Y deve essere compresa tra —32 e 209. Però, poiché un byte senza segno può assumere valori tra 0 e 255, nella tabella viene usata la notazione in complemento a 2 per le coordinate negative (troverete delle spiegazioni sulla notazione in complemento a due nella sezione che riguarda il linguaggio Assembly dello Z-80).
Diverso è il caso della coordinata X che viene spostata di 32 pixel a sinistra nel caso sia stato impostato a 1 il "clock early bit".
Per esempio, per posizionare uno sprite blu sul piano 0 con numero di modello 1 alle coordinate 100,100, le istruzioni che seguono costituiscono una alternativa all'uso di PUT SPRITE:

```
VPOKE 6912,100:VPOKE 6913,100

VPOKE 6914,1:VPOKE 6915,4
```

Per spostare a sinistra lo sprite di 32 pixel si potrebbe usare sia:

```
VPOKE 6913,68
```

sia: VPOKE 6915,132.

Con la seguente si possono togliere tutti gli sprite dallo schermo:

```
VPOKE 6912,208
```

Nel nostro prossimo esempio viene illustrato come copiare l'insieme dei caratteri maiuscoli nell'area dei modelli degli sprite per ottenere un testo in formato non standard:

```
10 SCREEN1,1:REM sprite espansi 8*8
20 FOR X=0 TO 207
30 VPOKE 14856+X,VPEEK(520+X)
40 NEXT
```

I numeri di modello dal 65 in su sono stati usati deliberatamente allo scopo di mantenere una corrispondenza numerica tra il numero del modello di sprite e il codice del carattere. Ciò permette infatti di visualizzare con facilità delle stringhe:

```
10 A$="MSX":FOR X=1 TO 3
20 PUT SPRITE X,(18*X,40),11,ASC(MID$(A$,X,1))
30 NEXT
```

I due restanti comandi per la manipolazione degli sprite sono:

1. ON SPRITE GOSUB
2. SPRITE ON

consentono di rilevare e gestire collisioni tra sprite. Una collisione si verifica quando si sovrappongono parti non trasparenti di due sprite.
Vi sono due limitazioni sull'informazione che è possibile dedurre da ogni collisione:

1. Si possono rilevare solo collisioni sprite-sprite e non tra sprite e sfondo.
2. Non è disponibile né una funzione BASIC né un registro del VDP che indichi quali sono i piani degli sprites coinvolti nella collisione.

L'istruzione SPRITE ON attiva la procedura di controllo che quindi viene eseguita dopo ogni istruzione BASIC. Nessun controllo viene effettuato nella modalità ad esecuzione diretta del BASIC. Quando avviene una collisione, viene eseguita un'istruzione GOSUB alla routine che è stata specificata nell'istruzione ON SPRITE GOSUB.
L'istruzione SPRITE OFF annulla ogni controllo riferito alle collisioni tra sprite. L'istruzione SPRITE STOP sospende l'effettiva attivazione della routine di gestione della collisione, ma il controllo viene mantenuto attivo. Infatti se avviene una collisione, essa viene 'ricordata' fino a quando viene eseguita un'istruzione di SPRITE ON nella chiamata della subroutine.
Un'istruzione di SPRITE STOP viene automaticamente eseguita ogni volta che si inizia l'esecuzione della routine di gestione delle collisioni. Analogamente quando si giunge al RETURN, viene eseguita una SPRITE ON - a meno che la routine stessa non contenga un'istruzione di SPRITE OFF.
È importante ricordare che l'istruzione CLEAR effettua una SPRITE OFF.
Accade spesso che il movimento degli sprite sia di natura fissata e ripetitiva. Vale a dire che la loro posizione deve solo essere incrementata e decrementata ad intervalli di tempo prefissati. Il BASIC MSX comprende un potente insieme di comandi per far sì che, durante l'esecuzione di un programma, ad intervalli di tempo regolari, venga effettuato un salto ad una determinata routine. L'intervallo di tempo viene specificato in multipli di 1/50 di secondo (cioè la frequenza tipica di interruzione del VDP).
Il formato dei suddetti comandi è simili a quello dei comandi per la gestione delle collisioni:

1. INTERVAL ON
2. ON INTERVAL=X GOSUB

Anche in questo caso esistono le varianti INTERVAL OFF e INTERVAL STOP; quest'ultima viene eseguita all'ingresso della routine di gestione. Se si specificano intervalli molto brevi, può verificarsi un leggero allungamento dell'intervallo effettivo a causa del controllo che viene effettuato dopo l'esecuzione di ogni istruzione BASIC.
Due gruppi di comandi simili a quelli fin qui illustrati sono:

1. ON STOP GOSUB e STOP ON/OFF/STOP
2. ON KEY GOSUB linea x, linea y ... e KEY (x) ON/OFF/STOP

La struttura ON STOP GOSUB / STOP ON va usata con una certa cautela poiché può impedire il ritorno dalla modalità BASIC di esecuzione di un programma a quella di esecuzione diretta dei comandi. Per esempio:

```
10 ON STOP GOSUB 40:STOP ON
20 PRINT "NOSTOP":GOTO20
40 RETURN
```

è un programma che non può essere interrotto. Infatti quando si vuole che l'esecuzione di un programma non possa essere interrotta, è sufficiente che la linea 10 appaia dopo un'istruzione CLEAR.

## ESEMPIO DI PROGRAMMA PER DISEGNARE SPRITE

Viene di seguito riportato un utile programma che vi permette di disegnare degli sprite a vostro piacimento.

```
 10 '*************************************
 20 '****      DISEGNO DI SPRITE       ****
 30 '*************************************
 40 '
 50 'USARE IL CURSORE PER MUOVERSI NELLA MATRICE
    DI DEFINIZIONE.
 60 '
 70 'BARRA SPAZIATRICE PER ALTERARE IL VALORE DEL BIT
    ALLA SINISTRA DEL CURSORE. CTRL/STOP PER TERMINARE
    E PER STAMPARE I VALORI DEI BYTE.
 80 '
 90 KEYOFF:FORX=0TO32:VPOKEX+14336,0:NEXT
100 ON STOP GOSUB300:STOP ON1
110 SCREEN1,2:COLOR14,1,1:CLS1
```

```
120 A$=STRING$(16,CHR$(250)):FORX=4TO19:LOCATE11,X:
    PRINTA$:NEXT
130 X=12:Y=4:LOCATEX,Y,1:PUT SPRITE0,(20,100),14,0
140 Q=0:ON STRIG GOSUB 250:STRIG(0) ON
150 FORL=1 TO 60:NEXT:D=STICK(0):ON D GOTO 170,150,190,
    150,210,150,230
160 GOTO 150
170 IF Y<>4 THEN Y=Y—1:LOCATEX,Y,1 ELSE 150
180 IF Y<>11 THEN 150 ELSE Q=Q—1:GOTO 150
190 IF X=27 THEN 150 ELSE X=X+1:LOCATEX,Y,1
200 IFX<>20 THEN 150 ELSE Q=Q+2:GOTO 150
210 IF Y=19 THEN 150 ELSE Y=Y+1:LOCATEX,Y,1
220 IF Y<>12 THEN 150 ELSE Q=Q+1:GOTO 150
230 IF X=12 THEN 150 ELSE X=X—1:LOCATEX,Y,1
240 IF X<>19 THEN 150 ELSE Q=Q—2:GOTO 150
250 IF VPEEK(6145+32*Y+X)=250 THEN VPOKE 6145+32*Y+X,
    219:AA=2 ELSE VPOKE 6145+32*Y+X,250:AA=4
260 BY=14336+Q*8:IF Q=0 OR Q=2 THEN BY=BY+Y—4 ELSE
    BY=BY+Y—12
270 IF Q=0 OR Q=1 THEN BI=19—X ELSE BI=27—X
280 VL=2^BI:IF AA=2 THEN VPOKEBY,(VPEEK(BY) OR VL) ELSE
    VPOKEBY,(VPEEK(BY) AND (255—VL))
290 RETURN
300 CLS: FORX=0TO15:LOCATE2,X+2:PRINTVPEEK(14336+X):
    LOCATE8,X+2:PRINTHEX$(VPEEK(14336+X)):LOCATE12,X+2:
    PRINTVPEEK(14352+X):LOCATE18,X+2:PRINTHEX$
    (VPEEK(14352+X)):NEXT
```

## Grafica ad alta risoluzione

Il BASIC MSX ha sette istruzioni che possono essere utilizzate solo nelle due modalità grafiche:

1) CIRCLE 2) DRAW 3) LINE 4) PAINT
5) PSET 6) PRESET 7) POINT

È ovvio però che queste istruzioni producono un risultato abbastanza scadente nella modalità multicolore dove la massima risoluzione è costituita da blocchi di 4 * 4 pixel.
Non è possibile visualizzare caratteri sullo schermo grafico usando le nor-

mali istruzioni. Viceversa i caratteri possono essere visualizzati come dati da file:

```
1. OPEN "GRP:"FOR OUTPUT AS#1
2. PRESET (40,40)
3. PRINT#1,"Testo grafico"
```

Si noti che, a meno che si sia previsto un numero maggiore di file con l'istruzione MAXFILES, il numero di file deve essere 1. Nella modalità multicolore ogni lettera è approssimativamente di 2,5 cm quadrati.
Il "cavallo di battaglia" delle istruzioni per la grafica è la DRAW nel seguente formato:

```
DRAW "comandi a lettera singola"
```

I 13 comandi disponibili costituiscono quello che la Microsoft definisce un "micro linguaggio per la grafica". Tramite essi si possono tracciare linee secondo una specifico passo a sinistra, a destra, in alto, in basso o in diagonale dal punto di riferimento corrente. Può essere utilizzato uno qualsiasi dei 16 colori disponibili. L'istruzione in questione è più comprensibile se si considera un esempio:

```
DRAW "D40R40U40L40"
```

Si ottiene un quadrato tracciando dapprima una linea di 40 unità di lunghezza verso il basso (D40), poi una verso destra (R40), poi una verso l'alto (U40) ed infine verso sinistra (L40). I principali comandi per determinare la direzione sono:

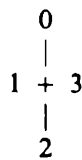
```
        U
    H       E
L       +       R
    G       F
        D
```

e "Cx" viene usata per impostare il colore.
Per tracciare una linea fino ad una posizione specificandone le coordinate, si deve usare M X,Y. In alternativa per muoversi fino ad una posizione di cui siano note le coordinate relative al punto di riferimento corrente, è possibile far precedere alla X e alla Y il segno "+" o il segno "—".
L'unità di misura dei movimenti può essere impostata dal comando Sn do-

ve n è compreso nell'intervallo 0 - 255 (default = 4). Dividendo questo numero per quattro si ottiene il numero dei pixel per ogni unità di spostamento. Se in un comando viene specificata una posizione sullo schermo, esterna all'intervallo —32768 +32767, non viene segnalato alcun errore e viene assunta la posizione più esterna possibile.
Esistono due varianti per ciascun comando di movimento:

1. Usando come prefisso al comando una lettera 'B' si provoca lo spostamento del cursore, senza che però venga tracciata alcuna linea.

2. Usando come prefisso 'N' si riporta il cursore grafico alla posizione iniziale, dopo il completamento dell'esecuzione del sottocomando.

Le direzioni effettive possono essere ruotate in senso antiorario per multipli di 90 gradi con il comando An, dove n deve essere compreso tra 0 e 3, come viene illustrato di seguito:

Quindi il comando DRAW "A3L40" serve per tracciare una linea verso destra per quaranta unità di spostamento.
Il sottocomando X consente d'includere una variabile stringa nella linea di comando.
La X precede il nome della variabile che viene seguito da un punto e virgola; per esempio:

```
ST$ = "M—20, + 40";DRAW"XST$;"
```

Questo sottocomando non è però necessario qualora non si usino i delimitatori: DRAW ST$. Inoltre anche una variabile numerica può essere usata in una linea di comando: è necessario farla precedere da un segno "=" e farla seguire da un ";": DRAW "U=Y;".
Il formato del comando LINE è il seguente:

```
LINE(X1,Y1)—(X2,Y2),colore
```

e provoca il tracciamento di una linea tra le due coordinate indicate; ciascuna coordinata può essere specificata in modo alternativo usando la variante STEP dell'istruzione.
Se il comando è fatto seguire da ",B" si ottiene il disegno di un rettangolo e se si usa ",BF" il rettangolo viene riempito

Il comando CIRCLE consente di ottenere diagrammi piuttosto sofisticati in modo semplice. Una qualsiasi ellisse può essere specificata e disegnata, interamente o parzialmente. Il formato del comando di base è:

```
CIRCLE (X,Y),raggio
```

con le opzioni:

```
colore, angolo di partenza, angolo finale, raggio apparente
```

Le coordinate X e Y specificano il centro del cerchio; usando il formato STEP (X,Y) è possibile definire tale punto relativamente al riferimento corrente. Quando si usano l'angolo di partenza e quello finale devono assumere valori tra 0 e 2 pi greco. Viene accettato anche un valore negativo, ma provoca la visualizzazione di ulteriori linee tra il centro dell'ellisse ed il punto di partenza e di arrivo. Il raggio apparente è il rapporto tra i raggi orizzontale e verticale.

Il comando PAINT è sufficientemente auto esplicativo; il suo formato è:

```
PAINT (X,Y) oppure PAINT(X,Y), colore
```

provoca il riempimento in colore di una qualsiasi figura che racchiuda le coordinate in questione. Se non viene specificato il colore, viene usato il colore corrente per il primo piano. Si noti che nella modalità HRG, il colore del bordo e quello specificato in PAINT devono essere i medesimi, altrimenti viene ricolorato l'intero schermo. Nella modalità multicolore si può usare la forma:

```
PAINT(X,Y), colore di riempimento, colore del bordo.
```

**Esempio:** *Per produrre un cerchio rosso pieno, nella modalità HRG, usare:*

```
SCREEN2:COLOR8,1,1:CIRCLE(80,80),40:PAINT(80,80)
```

I due restanti comandi:

```
PSET(X,Y), colore
PRESET(X,Y), colore
```

sono identici nel senso che entrambi servono per indicare un colore per il pixel specificato. Tralasciando il parametro colore, PSET usa il colore di primo piano, mentre PRESET usa il colore di sfondo.

PROGRAMMA DI ESEMPIO "SKETCH PAD"

Il seguente programma illustra l'uso della grafica ad alta risoluzione:

```
10 '************************************
20 '****          SKETCH PAD        ****
30 '************************************
40 '
50 'JOYSTICK NELLA PORTA 1. DISEGNARE SCHIACCIANDO
   IL PULSANTE DI ACCENSIONE
60 '
70 'F1 PER CAMBIARE COLORE; FAR SEGUIRE IL NUMERO
    DEL COLORE - DUE CARATTERI
80 'F2 PER TRACCIARE UN CERCHIO; FAR SEGUIRE DAL RAGGIO -
   DUE CARATTERI
90 'F3 PER RIEMPIRE UNA SAGOMA; DEVE ESSERE CHIUSA
   E DELLO STESSO COLORE USATO AL MOMENTO
100 'F4 PER SALVARE IL DISEGNO; DUE MINUTI PER IL
   COMPLETAMENTO DELL'OPERAZIONE.
110 'F5 PER CARICARE LO SCHERMO DA NASTRO; PREMI PLAY
   DOPO F5, DUE MINUTI PER CARICARLO.
120 '
130 FOR X = 38000!TO38024!:READA$:A = VAL("&H" + A$):
    POKEX,A:NEXT:DEFUSR = 38000!
140 DATA21,00,00,11,40,9C,01,00,18,CD,59,00,21,00,20,11,40,B4,
    01,00,18,CD,59,00,C9
150 FOR X = 38200!TO38224!:READA$:A = VAL("&H" + A$):POKEX,A:
    NEXT:DEFUSR2 = 38200!
160 DATA 21,40,9C,11,00,00,01,00,18,CD,5C,00,21,40,B4,11,00,20,
    01,00,18,CD,5C,00,C9
170 ON KEY GOSUB 350,360,370,380,410:KEY(1) ON:KEY(2)
    ON:KEY(3) ON:KEY(4) ON:KEY(5) ON
180 ON STRIG GOSUB 180,240:STRIG(1) ON
190 COLOR 10,1,10:SCREEN 2,0:CLS:X = 125:Y = 95
200 FOR Z = 14336 TO 14343:READ ZZ:VPOKEZ,ZZ:NEXT:C = 10
210 DATA 0,32,32,248,32,32,0,0,0
220 PUT SPRITE0,(X—2,Y—4),10,0:IF A = 1 THEN RETURN ELSE A = 1
230 GOTO 230
240 DN = STICK(1):IF DN = 0 THEN 240
250 ON DN GOTO 260,270,280,290,300,310,320,330
260 IF Y = 0 THEN 240 ELSE Y = Y—1:GOTO 340
270 IF X = 255 OR Y = 0 THEN 240 ELSE X = X + 1:Y = Y—1:GOTO 340
```

```
280 IF X=255 THEN 240 ELSE X=X+1:GOTO 340
290 IF X=255 OR Y=191 THEN 240 ELSE X=X+1:Y=Y+1:
    GOTO 340
300 IF Y=191 THEN 240 ELSE Y=Y+1:GOTO 340
310 IF X=0 OR Y=191 THEN 240 ELSE X=X—1:Y=Y+1:GOTO 340
320 IF X=0 THEN 240 ELSE X=X—1:GOTO 340
330 IF Y=0 OR X=0 THEN 240 ELSE Y=Y—1:X=X—1
340 IF C<>0 THEN PSET(X,Y),C:GOTO 220 ELSE 220
350 PUT SPRITE 2,(20,20),10,0:PUT SPRITE 4,(28,20),10,0:
    A$=INPUT$(1):PUT SPRITE 2,(0,0),,0:B$=INPUT$(1):PUT
    SPRITE 4,(0,0),,0:C=VAL(A$+B$):RETURN
360 PUT SPRITE 2,(20,80),10,0:PUT SPRITE 4,(28,80),10,0:
    A$=INPUT$(1):PUT SPRITE 2,(0,0),,0:B$=INPUT$(1):PUT
    SPRITE 4,(0,0),,0:R=VAL(A$+B$):CIRCLE(X,Y),R,C:RETURN
370 PAINT (X,Y),C,C:RETURN
380 Z=USR(2)
390 SCREEN1:CLS:LOCATE2,2:PRINT"PREMERE PLAY E RECORD
    QUINDI RETURN"
400 A$=INPUT$(1):IF ASC(A$)<>13 THEN 400 ELSE BSAVE
    "CAS:",40000!,52288!:STOP
410 PUT SPRITE 2,(20,20),10,0:BLOAD"CAS:":Z=USR2(2):PUT
    SPRITE 2,(0,0),,0:RETURN
```

## Generazione di suoni

Per ottenere una semplice segnalazione sonora è sufficiente usare l'istruzione BEEP o stampare CHR$(7). Suoni più complessi possono essere generati per mezzo dell'istruzione PLAY e del macro linguaggio ad essa collegato o andando direttamente a modificare il contenuto dei registri del PSG con il comando SOUND. Quest'ultimo ha il seguente formato:

SOUND numero di registro, valore

Per conoscere in dettaglio le funzioni dei registri si consulti anche la sezione dedicata all'AY-3-8910

L'istruzione PLAY può essere seguita da un massimo di tre stringhe, ciascuna della quali è una sequenza di comandi di una lettera che riguardano una 'voce' del PSG (il formato è simile a quello di DRAW).

Per esempio, per suonare una singola nota con la voce numero 2, si deve usare la seguente istruzione:

PLAY "","N20"

Poiché per molti parametri esiste un valore che viene assunto per default, non è necessario procedere ad una lunga fase di inizializzazione.
Esistono due metodi per indicare quale nota deve essere suonata:

1. Nx dove x è compreso nell'intervallo tra 0 e 96. 0 provoca una pausa musicale.

2. Impostando l'ottava tramite Ox (con x tra 1 e 8, 4 per default) e la nota con una lettera tra A e G (A=la, B=si, C=do, ecc., G=sol). Per esempio:

PLAY"O6ABCD" farà suonare la scala la-si-do-re della sesta ottava.

Se fosse seguita da:

PLAY "BCD" anche queste note verrebbero considerate della sesta ottava.

Diesis e bemolli (solo quelli disponibili sul pianoforte) vengono prodotti usando # o + dopo la nota per i diesis, e -- per i bemolli.

Il volume di default è 4, in una scala da 0 a 15, e può essere impostato con il sottocomando Vx.
La durata di ciascuna nota viene impostata usando il comando Lx dove x è compreso tra 1 e 64 (estremi inclusi). Il valore uno produce una nota della durata di una battuta musicale, il valore 4 una nota di un quarto e così via. Per ottenere una nota più lunga di 1 è necessario variare il tempo impostandolo con il comando Tx (default = 120). In quest'ultimo caso la x può assumere valori compresi tra 32 e 255 e determina il numero di note da un quarto che vengono suonate in un minuto. Per esempio, per fare in modo che tutte le note durino il doppio è necessario dare T60.
Se è necessario cambiare solo la lunghezza di una singola nota, si può mettere il reciproco della nota dopo la nota stessa, senza la "L": ad esempio B#8 o D32.
Si noti che con tutti i comandi di una sola lettera, si possono usare delle variabili internamente al comando:

```
PLAY "N=X;L=Y;N=X;"
```

Per semplificare la trascrizione la Microsoft ha reso possibile far seguire una nota da un punto per aumentarne la durata di metà. Un metodo alternativo a N0 per ottenere una pausa consiste nell'uso del comando Rx dove x è compreso tra 1 e 64 e viene interpretato nello stesso modo del comando L.
I due restanti sottocomandi di una sola lettera sono Mx e Sx: essi consentono di ottenere effetti sonori ancora più sofisticati modificando il timbro (l'inviluppo) del suono prodotto. Ogni forma d'onda inizia con una delle due seguenti sequenze:

1. Il volume cresce da 0 fino al massimo. In questo caso il timbro ha un valore di 'attacco' pari a 4.

2. Il volume decresce dal massimo fino a zero. Il timbro ha un valore di attacco di zero.

Il resto della forma dell'inviluppo è determinato da altri tre parametri. Il valore di ciascuno di questi parametri viene aggiunto a quello iniziale di attacco. Il totale viene usato con il comando S in modo da determinare il timbro del suono nel modo scelto:

1. Decadimento: se il suono deve terminare dopo il primo ciclo, questo parametro vale 0, altrimenti viene impostato a 8.
2. Tenuta: se impostato a 1, la sequenza di attacco viene ripetuta per ogni ciclo. Se impostato a 0, il volume viene mantenuto pari a quello che si aveva alla fine del primo ciclo.
3. Rilascio: il valore 2 fa sì che il volume sia modificato alla fine di ogni ciclo. Il valore 0 non provoca alcun cambiamento.

Ognuno dei suddetti parametri può essere impostato in due modi, e quindi sono possibili 16 varianti nella scelta della forma dell'inviluppo e della sua interazione con il ciclo. Si tratta però di una duplicazione poiché in realtà esistono solo otto possibili modelli diversi. Questi ultimi vengono riportati, insieme con i possibili valori dei parametri, nella quinta pagina dell'Appendice E. I valori di default sono:

Forma: 1
Modulazione: 255

Per esempio, per produrre note che mantengano il volume massimo raggiunto per tutta la loro durata, è necessario dare ai parametri i seguenti valori:

1. Attacco: 4
2. Decadimento: 8
3. Tenuta: 1
4. Rilascio: 0

Totale = 13 quindi PLAY"S13....

La durata del primo e dei cicli seguenti viene precisata con il comando M. Esso accetta valori interi compresi tra 1 e 65535 compreso.
Un'ultima notazione a proposito del comando PLAY è che (come il comando DRAW) al suo interno possono essere usate stringhe predefinite, con il sottocomando X:

```
A$="N40N20N40":PLAY"XA$;"
```

Se si usa una variabile essa deve essere seguita da un punto e virgola. Lo

stato di uno o di tutti i canali può essere ottenuto con il seguente formato dell'istruzione PLAY:

```
PLAY (x)
```

In questo caso x deve essere compreso tra 0 e 3. Se si controlla il canale 1, o il 2, o il 3, viene ritornato il valore —1 se il canale stesso è attivo, zero altrimenti. Se invece si fa eseguire PLAY(0) viene ritornato —1 se almeno uno dei tre canali è attivo, 0 altrimenti.

## Memorizzazione del programma

In tutti gli elaboratori MSX con almeno 32K di RAM, la memorizzazione di un programma BASIC inizia dalla locazione 32768 (&H8000). Ogni linea di programma non viene memorizzata così come viene introdotta, ma in una forma condensata. La parole chiave sono sostituite con numeri simbolo (tokens) mentre i nomi delle variabili e i simboli vengono memorizzati direttamente. Inoltre ogni linea viene preceduta da due coppie di byte. La prima coppia contiene l'indirizzo della linea successiva mentre la seconda contiene il numero della linea stessa. Le linee sono separate da un byte nullo (a zeri binari) e la fine del programma viene indicata con altri due byte con valore zero.
Appena oltre il programma c'è una tabella per le variabili. In essa vengono memorizzate tutte le variabili non vettori del programma. Infine segue una tabella per i vettori.
Per esempio, se viene introdotto il seguente programma:

```
10 FOR X=32768! TO 40000!
20 PRINTX,PEEK(X)
40 NEXT
```

la sua memorizzazione è organizzata nel modo seguente:

| *Locazione* | *Valore* | *Commento* |
|---|---|---|
| 32768 | 0 | |
| 32769 | 23 | Indirizzo della |
| 32770 | 128 | riga successiva |
| 32771 | 10 | Numero di riga |
| 32772 | 0 | |

| | | |
|---|---|---|
| 32773 | 130 | Sta per FOR |
| 32774 | 32 | Codice di spazio |
| 32775 | 88 | X |
| 32776 | 239 | Sta per = |
| 32777 | 29 | S.P.Costante |
| 32778 | 69 | S.P.Byte dell'esponente |
| 32779 | 50 | |
| 32780 | 118 | S.P.Mantissa |
| 32781 | 128 | |
| 32782 | 32 | Codice di spazio |
| 32783 | 217 | Sta per TO |
| 32784 | 32 | Codice di spazio |
| 32785 | 29 | S.P.Costante |
| 32786 | 69 | S.P. Byte dell'esponente |
| 32787 | 64 | |
| 32788 | 0 | S.P.Mantissa |
| 32789 | 0 | |
| 32790 | 0 | Demarcatore di fine riga |
| 32791 | 36 | Indirizzo della riga successiva |
| 32792 | 128 | |
| 32793 | 20 | Numero di riga |
| 32794 | 0 | |
| 32795 | 145 | Sta per PRINT |
| 32796 | 88 | X |
| 32797 | 44 | Virgola |
| 32798 | 255 | Sta per PEEK |
| 32799 | 151 | |
| 32800 | 40 | ( |
| 32801 | 88 | X |
| 32802 | 41 | ) |
| 32803 | 0 | Demarcatore di fine riga |
| 32804 | 42 | Indirizzo della riga successiva |
| 32805 | 128 | |
| 32806 | 40 | Numero di riga |
| 32807 | 0 | |
| 32808 | 131 | Sta per NEXT |
| 32809 | 0 | Demarcatore di fine riga |
| 32810 | 0 | Demarcatore di fine programma |
| 32811 | 0 | |

Notate come le costanti siano espresse in formato esteso. I parametri per il controllo del ciclo FOR..NEXT sono memorizzati in precisione semplice.

La tabella delle variabili semplici viene spostata quando si aggiungono o cancellano linee al programma. Le variabili sono memorizzate nell'ordine in cui esse appaiono nel programma. Il primo byte di ciascuna ne indica il tipo:

8. Doppia precisione.
4. Precisione semplice.
3. Stringa.
2. Intero.

I due byte che seguono sono i codici dei primi due caratteri del nome della variabile. La sezione restante è differente a seconda del tipo della variabile:

1. Intero: il valore viene memorizzato in forma binaria rovesciata, con segno e su due byte.

2. Stringa: tre byte vengono usati per memorizzare un dato di tipo stringa. Il primo specifica il numero di caratteri della stringa. Gli altri due contengono l'indirizzo attuale della stringa vera e propria. Questo indirizzo ha il byte alto e quello basso invertiti.

3. Precisione semplice: il valore viene rappresentato da un byte per l'esponente e da una mantissa a sei cifre, codificata in binario su tre byte. Il bit più significativo (m.s.b.) della mantissa indica, quando è a 1, che il valore è negativo.

4. Doppia precisione: uguale al caso della precisione semplice, solo che per la mantissa vengono usati sette byte.

Si noti che il byte dell'esponente, la cui locazione viene ritornata dalla funzione VARPTR, contiene un valore incrementato di &H40.
Ciascuna entrata della successiva tabella dei vettori inizia nuovamente con tre byte che contengono il tipo della variabile e il nome. Però prima dei dati veri e propri c'è una testata. Quest'ultima ha tre sezioni:

1. I primi due byte che contengono il numero dei byte restanti che compongono il vettore.

2. Un singolo byte che contiene il numero di dimensioni.

3. Una sequenza di valori su due byte che specificano l'ampiezza di ciascuna dimensione.

# Capitolo 3
# VOCABOLARIO DEL BASIC MSX

ABS(Y)

Funzione che ritorna il valore assoluto dell'espressione numerica Y.
Esempio: ABS(—4) ritorna 4.

ASC(Y$)

Ritorna il codice ASCII del primo carattere della stringa Y$. Se si tratta di un simbolo grafico, viene restituito 1. In caso di stringa nulla, viene segnalato un errore. Si faccia riferimento all'Appendice A per avere una lista completa dei codici dei caratteri.
Esempio: ASC("ABC") ritorna 65.

ATN(Y)

Funzione che ritorna il valore dell'arcotangente dell'espressione Y. Il risultato è in radianti, compreso nell'intervallo tra —PI greca/2 e PI greca/2. Se Y è una variabile, può essere di un tipo qualsiasi: comunque il calcolo viene effettuato in doppia precisione.
Esempio: C=ATN(40) ritorna 0.67474094222354

AUTO

Varianti: AUTO
AUTO numero di linea
AUTO ,incremento
AUTO numero di linea, incremento

I valori di default sono: numero di linea 0 nel primo caso, incremento 10 nel secondo e numero di linea 0 ed incremento 10 nel terzo caso.
Il successivo numero di linea viene visualizzato dopo ogni carattere di "Ritorno carrello" (Carriage Return o <CR>). Le opzioni consentono di specificare il numero di linea iniziale e l'incremento. La sequenza viene terminata con un CTRL-C o con un CTRL-STOP. Viene visualizzato un asterisco dopo ogni numero di linea che già esiste: in questo caso un ulteriore <CR> lascia inalterata la linea già esistente.
Esempio: AUTO 200,20 dà ia sequenza 200, 220, 240....

## BASE(Y)

Si tratta di una speciale variabile in cui vengono ritornate le locazioni delle tabelle nella RAM video usate dal VDP per produrre l'attuale schermata. I valori di Y compresi tra 0 e 19 restituiscono la posizione di partenza di una singola tabella per una data modalità di visualizzazione. Per ulteriori dettagli vi consulti l'Appendice C. Ciascun modo ha una tabella dei nomi e una tabella per la generazione dei modelli. Queste ultime contengono i dati del piano di visualizzazione.
Con l'eccezione della modalità testo a 40 colonne, i restanti modi hanno anche una tabella dei modelli degli sprite e una tabella per gli attributi degli sprite. La posizione di ciascuna di queste tabelle non cambia a seconda della modalità di visualizzazione:

Tabella degli attributi degli sprite: 6912
Tabella dei modelli degli sprite: 14336

Esempio: BASE(0) ritorna 0, cioè la posizione di partenza della tabella dei nomi nella modalità testo.

## BEEP

Genera un breve avvertimento sonoro.

## BIN$(Y)

Funzione che ritorna una stringa di caratteri "0" e "1" che costituiscono la rappresentazione binaria dell'espressione decimale Y.
Y deve essere compreso nell'intervallo —32768 e 65535. Un valore negativo viene espresso nella notazione in complemento a due.
Esempio: BIN$(—1) ritorna "1111111111111111"

BLOAD

Varianti: BLOAD"CAS:"
BLOAD"CAS:nome file"
BLOAD"CAS:",R
BLOAD"CAS:", offset

Carica un programma in codice macchina da cassetta (l'unico dispositivo previsto dalla versione 1 del BASIC), mettendolo in memoria alla locazione dalla quale era stato salvato.
L'opzione R esegue una chiamata, al termine del caricamento, alla locazione che era stata specificata alla BSAVE (il programma viene eseguito). L'opzione "offset" consente di specificare uno spostamento dell'indirizzo di caricamento del programma rispetto a quello da cui era stato salvato.
Esempio: BLOAD"CAS:",R,&H20 provocherà il caricamento e l'esecuzione del prossimo programma trovato sulla cassetta. Il programma verrà inoltre caricato 32 locazioni più in alto nella memoria rispetto alla posizione in cui era stato salvato.

BSAVE"CAS:",inizio, fine

Varianti: BSAVE"CAS:nome file", inizio, fine
BSAVE"CAS:", inizio, fine, indirizzo di attivazione

Si usa per salvare su cassetta l'immagine (cioè il contenuto corrente) di una porzione di memoria. È possibile specificare una locazione di attivazione qualora il programma sia poi caricato in memoria con una BLOAD"CAS:",R.
Se si usa BLOAD"CAS:",R senza che sia stata precisata una locazione in fase di salvataggio, l'attivazione avviene dalla prima locazione occupata.
Esempio: BSAVE"CAS:",34000,38000,36000 salva il blocco di 4K byte tra 34000 e 38000 (compresi). Se ora si carica il programma appena salvato con una BLOAD"CAS:",R, allora, al termine del caricamento, viene attivato il programma a partire dalla locazione 36000.

CALL istruzione

Varianti: CALL istruzione (argomento)
CALL istruzione (argomento, argomento...)

Nota: CALL può essere sostituito dal carattere di sottolineatura.
L'istruzione CALL è usata per eseguire un comando reso disponibile da

una cartuccia ROM di estensione.
Esempio: CALL SPSET(176,32)

CDBL(Y)

converte Y in un numero in doppia precisione. Y può essere di un qualsiasi tipo numerico.

CHR$(Y)

Viene ritornata una stringa di un solo carattere il cui codice ASCII è l'espressione Y. Per ottenere un simbolo grafico, è necessario prima eseguire CHR$(1).
Esempio: CHR$(65) ritorna A.

CINT(Y)

Converte Y, troncandolo, in un numero intero. Viene segnalato un errore se Y non è compreso tra —32768 e 32767.
Esempio: Y=4.8:?CINT(Y) dà come risultato 4.

CIRCLE (X,Y), raggio

Varianti: CIRCLE(X,Y), raggio, colore
CIRCLE(X,Y), raggio, colore, angolo partenza, angolo finale
CIRCLE(X,Y), raggio,, angolo partenza
CIRCLE(X,Y), raggio,,, angolo finale
CIRCLE(X,Y), raggio, colore, a.p., a.f., raggio apparente

Valori di default: 0 per l'angolo di partenza, 2*PI greco per quello finale.
Il centro del cerchio, usualmente indicato in coordinate assolute, può anche essere specificato relativamente ad una posizione di riferimento con l'opzione STEP(X,Y).
Se non viene specificato alcun colore, viene utilizzato quello corrente di primo piano. Se si usano le opzioni per l'angolo di partenza e per quello finale, i parametri devono essere in radianti, tra —2*PI greco e 2*PI greco.
Il raggio apparente determina il rapporto tra gli assi dell'ellisse.
Esempio; CIRCLE(20,20),10,10,3.142,2 produce una mezza ellisse da PI greco a 2*PI greco.

## CLEAR

Varianti: CLEAR
CLEAR spazio stringa
CLEAR spazio stringa, memtop

Default per il parametro memtop: &HF380
Tutti i file aperti vengono chiusi, le variabili azzerate e le stringhe rese nulle. Il valore di default per lo spazio stringa è 200 byte. Con il parametro "memtop" si indica la locazione di memoria più alta disponibile in BASIC. Il suo valore può venire abbassato qualora esista la necessità di allocare un codice macchina sull'area riservata al BASIC. In questo modo ci si assicura che il codice non sarà sovrascritto dai dati del BASIC.
Si noti che anche le istruzioni ON ... GOTO / GOSUB vengono annullate da questa istruzione.
Esempio: CLEAR 400,44000 fa sì che lo spazio riservato per le variabili stringa sia di 400 byte e che il massimo indirizzo occupabile dal BASIC sia 44000.

## CLOAD

Variante: CLOAD "nome file"

Questa istruzione chiude tutti i file e azzera tutte le variabili prima di procedere al caricamento del prossimo file di programma da cassetta. Se si indica il nome di un file, solo i primi sei caratteri sono significativi.

## CLOAD?

Variante: CLOAD? "nome file"

Il programma in memoria viene confrontato con il prossimo file di programma su cassetta (o quello indicato con "nome file"). Se vengono riscontrate differenze viene segnalato "Verify error".

## CLOSE

Varianti: CLOSE
CLOSE #Y
CLOSE #Y, #Z

Nota: il simbolo di diesis è opzionale.
A meno che venga specificato il numero di un file, tutti i file vengono chiusi e i buffer ad essi associati vengono rilasciati. Qualsiasi dato sia contenu-

to al momento dell'istruzione in un buffer di output viene prima visualizzato o stampato.
Esempio: CLOSE 4 chiude e rilascia ogni buffer associato al canale 4.

CLS

Pulisce lo schermo, lasciandolo del colore dello sfondo specificato dall'istruzione COLOR. Nella modalità testo, il cursore viene inoltre riportato nell'angolo a sinistra in alto. CLS è un'istruzione valida in tutte le modalità di visualizzazione.

COLOR sfondo, primo piano, bordo

Varianti: COLOR, sfondo
COLOR, sfondo, bordo
COLOR,, bordo
COLOR primo piano,, bordo

Quest'istruzione è valida in tutti i modi di visualizzazione. In entrambe le modalità testo, il colore di sfondo viene assunto immediatamente dopo la sua esecuzione. Nei modi ad alta risoluzione grafica e multicolore il cambiamento viene reso effettivo da un CLS. Nella modalità testo a 40 colonne il bordo assume il colore dello sfondo. In tutte le modalità di visualizzazione viene aggiornata l'intera tabella dei colori della VRAM.
Esempio: COLOR 4,10,14 farà in modo che il bordo sia grigio (14), il piano di visualizzazione giallo scuro (10) e il primo piano (cioè "l'inchiostro" con cui vengono scritti i caratteri) blu scuro (4).

CONT

Fa riprendere l'esecuzione di un programma dopo uno stop.

COS(Y)

Ritorna il coseno dell'angolo specificato, in radianti, dall'espressione Y.

CSAVE "nome file"

Variante: CSAVE "nome file", baud rate

Il programma attualmente in memoria viene salvato su cassetta. Sono significativi solo i primi sei caratteri del nome del file. Il file può essere sal-

vato sia a 1200 baud (valore di default), sia a 2400:

CSAVE "nome",1 .... 1200 baud
CSAVE "nome",2 .... 2400 baud

CSGN(Y)

Converte Y in un numero in singola precisione.

CSRLIN

Ritorna la posizione verticale del cursore sullo schermo, con valori da 0 a 23.
Esempio: RIGA=CSRLIN

DATA

Serve per dichiarare una serie di dati che verranno acceduti dal programma con l'istruzione READ. Su una linea è possibile avere sia dati numerici sia dati di tipo stringa. I valori devono essere separati da una virgola. Non è necessario racchiudere le stringhe tra doppi apici, a meno che esse non contengano virgole, due punti, punti e virgola o spazi significativi.
Esempio: DATA1.4,6,8.0,,,,4,2

DEF FN Y(A) = Espressione

Variante: DEF FN Y(A,B,C) = Espressione

In questo modo viene definita una funzione i cui parametri sono di solito indicati tra parentesi. Dopo che sono stati valutati, i valori tra parentesi vengono sostituiti nell'espressione. Questa deve essere dello stesso tipo dei parametri della definizione. Nella definizione stessa sono consentiti fino a un massimo di otto parametri.
Esempi: DEF FN Y(S) = 2 * S: T = 4: W = FN Y(T) implica che W assuma valore 8.
DEF FN Y(S,A) = 2*S + (A + 4):T = 2:R = 4:N = FN Y(T,R) implica che N assuma valore 2 * 2 + (4 + 4) = 12

DEFDBL Y
DEFINT Y
DEFSNG Y
DEFSTR Y

Varianti: DEFINT W-Z
DEFDBL A, B-S

Ciascuna delle istruzioni in questione viene usata per dichiarare il tipo di ogni variabile (semplice o vettore) che inizia con una determinata lettera. Questa definizione globale può essere sovrascritta per una singola variabile da una successiva dichiarazione di tipo.
Esempio: DEFINT A,B dichiara che tutte le variabili che iniziano con A o B saranno considerate di tipo intero.

DEFUSRx = indirizzo

Viene usata per indicare l'indirizzo di partenza di una routine in codice macchina. Questa routine verrà attivata per mezzo dell'istruzione USR. L'intero x deve essere compreso nell'intervallo tra 0 e 9. Se viene omesso viene assunto il valore 0.
Esempio: DEFUSR2 = &H4000

DELETE numero linea - numero linea

Varianti: DELETE numero linea
DELETE - numero linea

Cancella tutte la linee tra le due specificate. Il numero di linea che viene cancellato deve comunque esistere.

DIM Y(x)

Varianti: DIM Y(x), Z(n)

Se un vettore è composto da più di undici elementi, la presente istruzione serve per riservare dello spazio in memoria secondo le necessità. Un vettore di stringhe prevede l'allocazione di 255 byte per ciascun elemento.
Esempio: DIM Y(20),S$(12) consente di usare gli elementi da Y(0) a Y(20) e da S$(0) a S$(12).

DRAW "stringa di sottocomandi"

Si tratta del principale comando per la grafica; esistono tredici possibili sottocomandi. Per avere spiegazioni più dettagliate si faccia riferimento al Capitolo 2.

END

Termina l'esecuzione del programma e chiude tutti i file aperti. Non viene visualizzato alcun messaggio di "BREAK".

EOF(numero file)

Funzione che viene usata per controllare se è finito un file da cui si leggono dati in ingresso in modo sequenziale. Il valore ritornato è 0 tranne nel caso in cui il file sia effettivamente finito (ritorna —1).
Esempio: IF EOF(8)=—1 THEN 200

ERASE variabile di tipo vettore

Variante: ERASE vettore, vettore, ...

Cancella le variabili di tipo vettore il cui nome inizia con le lettere che vengono specificate. Per essere riusate, tali variabili devono essere nuovamente dimensionate.
Esempio: ERASE X,Y cancellerà XC(16) e YJ(14)

ERL

Quando si verifica un errore, in ERL viene memorizzato il numero di linea nella quale si è verificato l'errore stesso. Se ciò accade in modalità di esecuzione diretta, ERL contiene 65535.

ERR

Quando si verifica un errore, ERR ne contiene il codice corrispondente.

ERRORx

Istruzione che viene usata per forzare l'errore di cui viene specificato il codice. La x deve essere un intero compreso tra 1 e 255. Il BASIC usa codici di errore compresi tra 1 e 59: se vengono forzati provocano la stampa di messaggi esplicativi sul tipo di errore. Il messaggio "Unprintable error" (Errore non stampabile) viene prodotto nel caso in cui venga forzato un errore ed esso sia gestito dall'istruzione ON ERROR GOTO, ma non sia stata data alcuna definizione per quel codice di errore.
Esempio:

```
10 ON ERROR GOTO 200
:
:
100 ERROR 100
```

```
200 IF ERR=100 THEN END
```

### EXP(Y)

Viene ritornato il valore di "e" (la costante Neperiana) elevato alla Y. L'espressione Y deve assumere valori tra —147.3654459516 e 145.06286085862.

### FIX(Y)

Funzione che ritorna l'approssimazione intera dell'espressione Y. Essa differisce dalla INT per il trattamento delle espressioni negative. Infatti FIX tronca semplicemente il valore, mentre INT arrotonda, restituendo il valore intero più prossimo.
Esempi:
FIX(—2.8) ritorna —2
INT(—2.8) ritorna —3

### FOR Y=x1 TO x2 STEP x3

Tutte le istruzioni che seguono la suddetta, fino alla NEXT appropriata, sono ripetute per tutti i valori di Y compresi tra x1 e x2, considerati di volta in volta con un incremento pari a x3. Se viene omessa l'opzione STEP x3, l'incremento viene assunto unitario.
Se viene posto a zero il ciclo viene ripetuto indefinitivamente. Se x2 è minore o uguale a x1 e il passo è positivo, le istruzioni vengono eseguite una volta.

### FRE(" ")

Ritorna il numero dei byte restanti per la memorizzazione di stringhe.

### FRE(x)

Funzione che ritorna il numero di byte disponibili per la memorizzazione del programma. Poiché x è un argomento fittizio, esso può assumere un qualsiasi valore; inoltre non viene usato nella valutazione.
Esempio: Z=FRE(2)

### GOSUB / RETURN

Variante: RETURN numero di linea

Si tratta di una forma particolare dell'istruzione GOTO. Infatti quando viene incontrata l'istruzione RETURN, l'esecuzione riprende dalla linea successiva a quella in cui era l'istruzione GOSUB.

GOTO numero linea

Istruzione che trasferisce il controllo dell'esecuzione alla linea di cui viene indicato il numero (salto non condizionato).

HEX$(Y)

Funzione che ritorna l'equivalente in numerazione esadecimale dell'espressione intera Y. Questa deve essere nell'intervallo tra —32768 e 65535.
Esempio: HEX$(20) ritorna 14.

IF condizione THEN

Varianti: IF .. THEN .. ELSE ..
IF .. THEN .. IF .. THEN .. ELSE ..
IF .. GOTO
IF .. GOTO .. ELSE ..

Consente di eseguire un particolare gruppo di istruzioni solo se la condizione specificata è vera. Non è necessario usare GOTO per eseguire un salto dopo THEN e ELSE. Nella variante IF .. THEN .. IF .. THEN .. ELSE .. l'esecuzione prosegue con la linea seguente se la prima condizione non è soddisfatta.
Esempio: 10 X=0:IF X=1 THEN IF Y=2 THEN 40 ELSE 80 provocherà la continuazione dell'esecuzione dalla linea 20 (la successiva).

INP(indirizzo di una porta)

Un byte viene letto dalla porta specificata.

INPUT Y

Varianti: INPUT "stringa di avviso";Y
INPUT #numero di file, Y
INPUT$(x)
INPUT$(x, numero di file)

Quest'istruzione visualizza un punto di domanda per invitare l'utente al-

l'introduzione di un dato che verrà assegnato alla variabile indicata.
Se questa variabile è stata dichiarata prima dell'istruzione INPUT, e l'utente non introduce alcun dato prima di premere il <CR>, il valore della stessa non viene modificato; in caso contrario viene posto uguale a 0.
Le variabili possono essere semplici o vettori, sia di tipo stringa che numerico. Possono essere indicate più variabili separate da virgola.
Esempio: INPUT "X,Y$ <ENTER>";X,Y$ provocherà la visualizzazione del messaggio: X,Y$ <ENTER>? di seguito al quale l'utente introdurrà i valori richiesti.
La seconda variante prevede invece la possibilità di indicare il numero di un file come sorgente di dati. Le varianti 3 e 4 ritornano entrambe una stringa di x caratteri - la terza prendendoli dalla tastiera, e la quarta dal file indicato. In entrambi i casi non è necessario terminare l'introduzione del dato con un <CR>.

## INKEY$

Funzione che ritorna il primo carattere della coda associata alla tastiera; se questa è vuota viene ritornata una stringa nulla.
Esempio: 10 A$=INKEY$:IF A$="" THEN 10.
Si noti che lo stesso effetto di attesa del primo carattere battuto può essere ottenuto più efficientemente da A$=INPUT$(1).

## INSTR(A$;B$)

Variante: INSTR(x,A$,B$)

Ritorna la posizione della stringa B$ all'interno della stringa A$. Se B$ è una stringa nulla oppure se non è contenuta in A$ viene ritornato il valore 0. La variante consente di effettuare la ricerca a partire da x caratteri dall'estrema sinistra di A$.
Esempio: A=INSTR(2,"ABCD","D") fa sì che A assuma valore 4.

## INT(Y)

Ritorna il valore intero dell'espressione Y. Si veda, per confronto, anche l'istruzione FIX.
Esempio: INT(4.8) ritorna 4.

## INTERVAL ON /OFF/ STOP

Istruzione che attiva, disattiva o sospende la chiamata di un sottoprogramma

ad intervalli di tempo specificati dalla ON INTERVAL. Per maggiori dettagli si veda il capitolo due.

KEY numero di tasto funzionale,"stringa"

Associa la stringa data al tasto funzionale indicato. La stringa può essere lunga fino a un massimo di 15 caratteri.
Esempio: KEY 2,"RUN"+CHR$(13)

KEY LIST

Vengono listate, in ordine, le stringhe associate ai vari tasti funzionali.

KEY ON / OFF

Provoca la visualizzazione (o viceversa la cancellazione) dei tasti funzionali sulla ventiquattresima riga di testo dello schermo.

KEY (numero tasto funzionale) ON / OFF / STOP

Attiva, disattiva o sospende la chiamata a un sottoprogramma specificato dall'istruzione ON KEY GOSUB. Se viene attivata, il controllo che sia stato premuto il tasto funzionale in questione viene effettuato dopo l'esecuzione di ogni istruzione BASIC. Per maggiori dettagli si veda il Capitolo 2.

LEFT$(Y$,x)

Ritorna gli x caratteri più a sinistra della stringa Y$. Un carattere grafico occupa due posizioni. Se x è zero viene ritornata una stringa nulla. Se x è maggiore del numero di caratteri di Y$, viene ritornata Y$, senza che ad essa siano aggiunti degli spazi.
Esempio: ?LEFT$("NUOVA",2) ritorna NU.

LEN(Y$)

Funzione che ritorna il numero di caratteri - compresi quelli grafici e di controllo - della stringa Y$. Un carattere grafico è composto da CHR$(1) e dal codice vero e proprio del carattere.

LET variabile = espressione

Istruzione che viene usata per assegnare il valore di un'espressione ad una variabile. Nota: opzionale nel BASIC MSX.

LINE (x1,y1)—(x2,y2)

Varianti: LINE (x1,y1)—(x2,y2), colore
LINE (x1,y1)—(x2,y2), colore, B
LINE (x1,y1)—(x2,y2), colore, BF
LINE —(x2,y2), colore

Il formato STEP(X,Y) consente inoltre di sostituire ogni coordinata assoluta con coordinate relative al punto iniziale di riferimento.
Si tratta comunque di un'istruzione propria delle modalità grafiche per tracciare una linea tra i punti di cui vengono indicate le coordinate. L'opzione B disegna un rettangolo, mentre la BF disegna anch'essa un rettangolo ma lo riempie. Per maggiori dettagli si veda il Capitolo 2.

LINE INPUT Y$

Variante: LINE INPUT "messaggio"; Y$

Acquisisce un'intera linea da tastiera assegnandola alla variabile specificata. Questa istruzione non può essere usata sia nella modalità HRG (grafica ad alta risoluzione) sia in quella multicolore. Il numero di caratteri introdotti non deve esere superiore a 254. L'assegnazione effettiva alla variabile avviene quando si preme il <CR>.

LINE INPUT# numero del file, Y$

Assegna il contenuto di una linea, letta dal file sequenziale di cui viene indicato il numero, alla variabile stringa specificata.

LIST

Varianti: LIST numero di linea
LIST numero di linea -
LIST numero di linea - numero di linea
LIST - numero di linea
LIST .

Comando per avere una lista di tutte o di parte delle istruzioni del programma residente in memoria. L'opzione ".'' può essere usata per:

1. Listare nuovamente la linea precedentemente visualizzata.
2. Listare la linea che ha provocato l'interruzione del programma in esecuzione.

LLIST

Varianti: le stesse di LIST.

Quest'istruzione provoca la stampa effettiva su una stampante dell'intero programma corrente (o parte di esso).

LOAD "nome file"

Varianti: LOAD "riferimento al dispositivo"
LOAD "riferimento al dispositivo: nome file"
LOAD "riferimento al dispositivo: nome file", R

Il comando LOAD chiude tutti i file aperti e cancella il programma corrente dalla memoria, prima di caricarvi il file di programma, in formato ASCII, specificato. Se viene usata l'opzione R, i file non vengono chiusi e l'esecuzione riprende non appena è conclusa la fase di caricamento. Per una lista dettagliata dei riferimenti ai dispositivi si veda l'istruzione OPEN.

LOCATE colonna, riga.

Varianti: LOCATE ,riga
LOCATE ,, modo del cursore
LOCATE colonna, riga, modo del cursore

L'istruzione LOCATE può essere usata solamente nelle modalità testo e sposta il cursore alla colonna e riga indicate. Nella modalità a 40 colonne i valori consentiti sono:

Colonne: 0 - 36
Righe: 0 - 22/23 (dipende dal modo del cursore)

Nella modalità a 32 colonne:

Colonne: 0 - 28
Righe: 0 - 22/23 (dipende dal modo del cursore)

Le colonne non utilizzabili possono essere accessibili solo scrivendo (con VPOKE) nella VRAM - si veda a questo proposito il Capitolo 2. Il modo del cursore può essere:

0: il cursore non viene visualizzato

1: il cursore viene visualizzato

LOG(Y) con Y > 0

Funzione che ritorna il logaritmo naturale dell'espressione Y, che deve essere positiva.

MAXFILES = Y

Istruzione che serve per specificare il massimo numero di file che possono essere aperti contemporaneamente. L'espressione intera Y deve essere compresa tra 0 e 15. L'uso di quest'istruzione è necessario qualora si voglia aprire più di un file alla volta.

MERGE

Varianti: MERGE "nome file"
MERGE "riferimento al dispositivo"
MERGE "riferimento al dispositivo, nome file"

L'istruzione MERGE viene usata per unire e combinare il prossimo programma in formato ASCII presente su nastro con quello attualmente in memoria. Se i numeri di linea si sovrappongono, le linee iniziali vengono sostituite da quelle del secondo programma.
Esempio: per caricare ed unire al programma attualmente in memoria il prossimo programma su cassetta usare: MERGE"CAS:"

MID$(A$,x)=carattere

Variante: MID$(A$,x,n)= n caratteri

Istruzione che viene usata per sostituire l'x-esimo carattere di A$. Si ricordi che tutti i caratteri grafici sono preceduti da CHR$(1). La variante consente di specificare la modifica di n caratteri. Se x è maggiore della lunghezza di A$ viene ritornata una stringa nulla.
Esempio: A$="WAS":MID$(A$,1,2)=" I":?A$ implica che venga visualizzato " IS".

MOTOR

Varianti: MOTOR ON (condizione di default)
MOTOR OFF

Viene cambiato lo stato del motore dell'unità nastro per cassette.
Esempio: Se il motore dell'unità è fermo, il comando MOTOR lo metterà in movimento, e viceversa.

NEW

Il programma corrente viene cancellato dalla memoria e tutte le variabili vengono azzerate.

OCT$(Y)

Ritorna una stringa che rappresenta in numerazione ottale il valore dell'espressione decimale Y.

ON Y GOTO x1,x2, ... xn / ON Y GOSUB x1,x2, ... xn

Se Y=1 viene effettuato un salto (GOTO) o una chiamata a sottoprogramma (GOSUB) al primo numero di linea indicato nella lista: x1. Se Y=2 viene considerato in modo analogo il secondo elemento x2 e così via. Se l'espressione Y vale 0 oppure è maggiore del numero di elementi della lista, l'esecuzione del programma prosegue dalla linea immediatamente successiva.

ON ERROR GOTO x

Se si verifica un errore viene effettuato un salto alla linea x. Se x vale 0, la gestione degli errori viene normalizzata. La routine di gestione degli errori (quella che inizia all'istruzione x) viene terminata da un'istruzione di RESUME.

ON INTERVAL = Y GOSUB X

Per mezzo di quest'istruzione viene definito il sottoprogramma che verrà eseguito dopo ogni intervallo di tempo della lunghezza indicata (Y * 1/50 di secondo). Tale condizione deve essere attivata per mezzo di INTERVAL ON.

ON KEY GOSUB x1,x2, ... xn

Viene iniziata la chiamata a un sottoprogramma quando viene premuto un particolare tasto funzionale. Il primo numero di linea viene associato al tasto nmero 1, il secondo al numero 2 e così via. La chiamata avviene effettivamente se l'associazione è stata resa attiva da un'istruzione KEY(x) ON.
Esempio: ON KEY GOSUB 20,,,40 implica che non sono stati definiti i sottoprogrammi associati ai testi funzionali 2 e 3.

ON SPRITE GOSUB Y

Definisce il sottoprogramma che verrà attivato, qualora la condizione sia stata attivata da una SPRITE ON, alla collisione di due sprite.

ON STOP GOSUB Y

Definisce il sottoprogramma, che inizia alla linea Y, che verrà attivato quando verranno premuti simultaneamente il tasto STOP e quello di CTRL. L'azione viene effettivamente intrapresa se è stata preventivamente eseguita una STOP ON.

ON STRIG GOSUB x

Variante: ON STRIG GOSUB x1,x2,x3,x4

Identifica il sottoprogramma che viene attivato, qualora sia stata eseguita una STRIG ON (Y), quando viene premuta la barra di spaziatura.
La variante viene usata per consentire l'attivazione di diversi sottoprogrammi a seconda del segnale proveniente dai due joystick. Le linee di inizio dei sottoprogrammi devono essere fornite nel seguente ordine:

1. Barra di spaziatura
2. Segnale 1, Joystick 1
3. Segnale 1, Joystick 2
4. Segnale 2, Joystick 1
5. Segnale 2, Joystick 2

Esempio: STRIG (2) ON: ON STRIG GOSUB 200,400,600 implica che a fronte del segnale 1 del joystick 2 venga attivata la routine che inizia a linea 600.

OPEN"riferimento al dispositivo": AS numero di file

Varianti: OPEN "riferimento al dispositivo:nome file"
AS numero di file
OPEN "riferimento al dispositivo:nome file"
FOR modo AS numero file

Quest'istruzione apre un canale per un dispositivo e alloca un buffer di ingresso/uscita. Un file deve essere aperto prima che sia eseguita un'istruzione che faccia riferimento al numero di file ad esso associato. Per esempio

PRINT#, INPUT#, PUT o GET. Possono essere usati quattro riferimenti a dispositivi:

1. CAS: cassetta
2. LPT: stampante
3. CRT: video
4. GRP: schermo grafico

Il numero associato ad un file viene usato da numerose istruzioni per la gestione dell'ingresso/uscita; deve essere compreso tra 0 e Y, ove Y è un intero specificato per mezzo dell'istruzione MAXFILES.
La variante 'FOR modo' serve a precisare il tipo di trasferimento dei dati:

'INPUT': file di ingresso sequenziale
'OUTPUT': file di uscita sequenziale
'APPEND': file sequenziale a cui vengono fatte delle aggiunte in coda.

OUT numero della porta, byte dato

Serve per mettere il byte di dato sulla porta d'uscita di cui viene indicato il numero. Sia il numero di porta che il byte di dato devono essere compresi tra 0 e 255. La configurazione standard MSX non prevede più di 256 porte di ingresso/uscita.

PAD(Y)

Ritorna lo stato di un particolare tipo di periferica che può essere connessa ad un elaboratore MSX, la tavoletta tattile. Y può assumere valori tra 0 e 7:

0 - 3 per riferirsi alla tavoletta connessa alla porta 1.

4 - 7 per riferirsi alla tavoletta connessa alla porta 2.

Ciascuno dei quattro valori che possono essere scelti per ogni porta serve per avere lo stato di differenti parametri:

0 e 4 ritornano —1 se la tavoletta viene premuta, 0 se viene rilasciata.

1 e 5 ritornano la coordinata X del punto in cui la tavoletta è stata premuta.

2 e 6 ritornano la coordinata Y.

3 e 7 ritornano —1 se è premuto l'interruttore, 0 altrimenti.

PAINT (X,Y)

Varianti: PAINT STEP(X,Y)
PAINT (X,Y), colore
PAINT (X,Y), colore, colore della linea di bordo

Si tratta di un'istruzione che può essere usata sia nella modalità grafica ad alta risoluzione sia in quella multicolore. L'oggetto in cui è racchiusa la posizione specificata viene riempito con il colore dello sfondo. Si noti che nel modo HRG la linea del bordo deve essere sempre del colore di riempimento. Per maggiori dettagli si faccia riferimento al capitolo due.

PDL(Y)

Ritorna lo stato di un paddle connesso sia al terminale A sia al terminale B. L'espressione Y deve essere un intero compreso nell'intervallo tra 1 e 12. Se il valore è dispari, le informazioni vengono lette dal terminale A, viceversa dal B. Il valore ritornato è compreso tra 0 e 255.

PEEK(Y)

Ritorna il valore intero memorizzato all'indirizzo Y. L'espressione numerica Y deve essere compresa tra —32768 e 65535 (estremi compresi).

PLAY "stringa sottocomando"

Varianti: PLAY "stringa", "stringa", "stringa"
PLAY "stringa, " ", "stringa"
PLAY Y$,Z$,X$
PLAY "Y$;stringa", "stringa", "stringa"

Si tratta della principale istruzione per la generazione di suoni. Ciascuna delle tre stringhe si riferisce a un canale di uscita audio. Per maggiori dettagli si veda il Capitolo 2.

PLAY(Y)

Funzione che ritorna lo stato del canale (o dei canali) musicale specificato. Y deve assumere valori tra 0 e 3. Se nel buffer indicato vi sono dati, viene ritornato —1. Se il canale non è attivo viene ritornato 0. PLAY(0) ritorna lo stato di tutti e tre i canali.

POINT(X,Y)

Ritorna il codice di colore del pixel di cui vengono indicate le coordinate. Quest'istruzione ha effetto solo nelle due modalità grafiche. Vien ritornato —1 se una delle coordinate è esterna all'intervallo di validità.

### POKE X,Y

Scrive il valore di Y all'indirizzo X. Y deve assumere valori tra 0 e 255. L'intervallo di validità per l'indirizzamento è da —32768 a 65535. Un indirizzo negativo viene prima sommato a 65535.
Esempio: POKE 40000,254 scrive 254 all'indirizzo 40000.

### POS(Y)

Ritorna la posizione orizzontale del cursore in entrambe le modalità testo. Y è un argomento fittizio che non viene usato nella valutazione.

### PRESET(X,Y)

Varianti: PRESET (X,Y), colore
PRESET STEP(X,Y)

Il pixel dello schermo grafico di cui sono indicate le coordinate viene colorato con il colore dello sfondo. Le varianti consentono di specificare un colore diverso o d'indicare le coordinate relative ad un punto di riferimento prefissato.

### PRINT

Varianti: PRINT "stringa"
PRINT "stringa";
PRINT A$
PRINT Y
PRINT ,X$
PRINT CHR$(Y)+A$

Nota: PRINT può essere sostituito da "?".
Se la parola chiave non è seguita da alcuna espressione, viene stampata (visualizzata) una linea bianca. In ogni modo la stampa dell'ultima entità che segue PRINT viene sempre seguita da un <CR>, a meno che non sia a sua volta seguita, nell'istruzione, da un punto e virgola o da una virgola. Un punto e virgola provoca che l'entità venga stampata immediatamente dopo l'ultima visualizzata in precedenza, mentre la virgola fa sì che la stampa inizi dalla zona di tabulazione più prossima. Le zone di tabulazione sono ampie 14 caratteri. I caratteri di controllo possono essere usati per mezzo della funzione CHR$(Y).

PRINT # numero file, espressione

Vale quanto detto per PRINT, tranne che l'output avviene verso il canale indicato. Un <CR> <LF> (CHR$(13),CHR$(10)) vengono fatti seguire all'ultima entità della lista di output.

PRINT USING

Variante: PRINT # numero file, USING

Consente di stampare indicando esplicitamente il formato per mezzo di caratteri di controllo. Per maggiori dettagli si veda il Capitolo 2.

PSET (X,Y), colore

Variante: PSET STEP (X,Y), colore

Il pixel indicato viene colorato, sullo schermo grafico, nel modo voluto.

PUT SPRITE numero del piano di visualizzazione

Varianti: PUT SPRITE numero piano, (X,Y)
PUT SPRITE numero piano, STEP(x,y)
PUT SPRITE numero piano, (X,Y), colore
PUT SPRITE numero piano, (X,Y), colore, numero del modello

Si tratta del principale comando per la gestione degli sprite.
Con PUT SPRITE è possibile determinare la posizione, il colore e il modello per uno sprite. Il cursore di sprite viene posizionato sull'angolo dello sprite in alto a sinistra.
Un solo sprite può essere messo su ogni piano, ma un numero qualsiasi di sprite può assumere lo stesso modello. Per maggiori dettagli si veda il Capitolo 2 e l'istruzione SPRITE$.
Esempio: PUT SPRITE 0,(40,40),4,2

READ Y

Variante: READ Y,X,Z..

L'istruzione READ viene usata per assegnare le costanti di un'istruzione DATA alle variabili che vengono specificate. Il dato letto deve essere di tipo compatibile con la variabile a cui viene asssegnato.

## REM

Istruzione usata per introdurre dei commenti e delle note in un programma. Ogni successiva istruzione sulla stessa linea logica viene ignorata al momento dell'esecuzione, che invece prosegue da quella successiva. REM può essere sostituito da un singolo apice (').

## RENUM

Varianti: RENUM nuovo numero
RENUM nuovo numero, vecchio numero
RENUM nuovo numero, vecchio numero, incremento
RENUM vecchio numero di linea

Valori di default per il nuovo numero e per l'incremento: 10
Tutte le linee del programma vengono rinumerate, compresi i riferimenti che seguono le istruzioni GOTO, GOSUB, THEN, ELSE, ON .. GOSUB, IF THEN e ERL.
Il programma risultante inizia da linea 10 o, se viene indicato, dal nuovo numero. La variante con l'indicazione del "vecchio numero" fa sì che solo parte del programma, dal vecchio numero in avanti, venga effettivamente rinumerata.

## RESTORE

Variante: RESTORE numero di linea

Istruzione che fa in modo che il prossimo dato letto sia il primo della prima linea di istruzione DATA. Con la variante è possibile specificare una linea di istruzione DATA diversa da quella iniziale.
Esempio: RESTORE 200 farà in modo che la prossima istruzione di READ acceda all'istruzione DATA in linea 200.

## RESUME

Varianti: RESUME NEXT
RESUME numero linea

L'istruzione RESUME viene usata per terminare un sottoprogramma di gestione di un errore che sia stato attivato per mezzo di un'istruzione ON ERROR GOTO. Il suo effetto è quello di far continuare l'esecuzione del programma a partire dalla linea in cui si era verificato l'errore. La variante RE-

SUME NEXT è simile, tranne che l'esecuzione prosegue dalla linea successiva a quella che ha dato luogo alla situazione di errore.

RIGHT$(A$,x)

Funzione che ritorna una sottostringa di A$ contenente i suoi x caratteri più a destra.
Se x è più grande del numero di caratteri di A$, viene ritornato A$. Se x = 0, viene ritornata una stringa nulla. Un carattere grafico comprende sempre un prefisso di CHR$(1).
Esempio: RIGHT$("SINISTRA",20) ritornerà SINISTRA.

RND(Y)

Ritorna un numero casuale compreso tra 0 e 1, estremi esclusi.
Se Y > 0 viene generato il prossimo numero casuale secondo la sequenza interna alla funzione stessa.
Se Y = 0 viene restituito l'ultimo numero già generato.
Se Y < 0 il generatore di numeri casuali viene rinizializzato con Y come radice.
Si noti che la sequenza di numeri casuali prodotta da RND(—2) è la stessa prodotta da ogni successiva attivazione RND(—2). Per produrre una sequenza 'nuova' vi consigliamo di usare RND(—TIME).

RUN numero di linea

Esegue il programma BASIC attualmente in memoria a cominciare dalla linea di cui è indicato il numero. Se quest'ultimo non compare, l'esecuzione ha inizio dalla prima linea del programma.

SAVE "nome file"

Varianti: SAVE "riferimento al dispositivo:"
SAVE "riferimento al dispositivo:nome file"

Trasferisce un programma BASIC in formato ASCII sul dispositivo e/o sul file che vengono indicati. La fine del file viene codificata come CTRL-Z. Si noti che non è possibile verificare un file. La velocità di trasferimento viene specificata per mezzo dell'istruzione SCREEN.
Esempio: SAVE "CAS:PROG4"

SCREEN modo, dimensione degli sprite selettore di "keyclick", baud rate della cassetta, opzione stampante

L'istruzione SCREEN viene usata per selezionare sei opzioni nel modo seguente:

*Modo:* 0 Modalità testo 40 * 24
1 Modalità testo 32 * 24
2 Modalità ad alta risoluzione
3 Modalità multicolore

*Dimensione degli sprite:*
0 8 * 8 pixel
1 8 * 8 pixel ingranditi
2 16 * 16 pixel
3 16 * 16 pixel ingranditi

*Selettore di "keyclick":*
0 "keyclick" attivo (premendo un qualsiasi tasto sulla tasiera si sente un leggero "click")
1 "keyclick" disattivato

*Baud rate della cassetta:*
1 velocità di trasferimento dati pari a 1200 baud
2 2400 baud

valida sia per il formato dei programmi sia per il formato ASCII.

*Opzione stampante:*
0 Stampante MSX
<0> Assenza delle facilitazioni grafiche dell'MSX

Se viene selezionata l'ultima opzione, tutti i simboli grafici vengono sostituiti con degli spazi.

SGN(Y)

Se Y > 0 allora SGN ritorna 1, se Y = 0 allora viene ritornato 0, altrimenti —1.

SIN(Y)

Ritorna il seno dell'espressione Y considerata espressa in radianti.

SOUND registro, Y

L'istruzione SOUND scrive il valore dell'espressione Y nel registro specificato del PSG. Y deve essere compreso nell'intervallo 0 - 255, estremi compresi. Per maggiori dettagli si veda il capitolo sull'AY-3-8910.
Esempio: SOUND 4,4

SPACE$(Y)

Ritorna una stringa composta da Y spazi. Y compreso tra 0 e 255.

SPC(Y)

Viene usata in unione con le istruzioni PRINT e LPRINT al fine di produrre Y spazi. Y compreso tra 0 e 255.
Esempio: ?SPC(4);"SOVRASCRIVE"

SPRITE$(Y) = "stringa di definizione"

Variabile di sistema per le definizione del modello di uno sprite. Il codice carattere per ciascun carattere della stringa allocata è il modello di un byte dello sprite. La forma binaria consente una definizione di sprite più agevole:

CHR$(&10101011)

Se si stanno usando gli sprite a 16 * 16 pixel, possono essere definiti fino a 64 sprite (Y compreso tra 0 e 63). Quando invece si usano gli sprite quadrati 8 * 8 pixel si possono definire addirittura fino a 255 modelli. Per ulteriori dettagli si consulti il Capitolo 2.

SPRITE ON /OFF / STOP

Attiva, disattiva o sospende la chiamata di un sottoprogramma all'avvenire di una collisione tra due sprite. L'istruzione ON SPRITE GOSUB deve essere stata precedentemente usata per specificare quale routine debba essere attivata.
Esempio: SPRITE STOP sospenderà l'attivazione di sottoprogrammi in occasione della collisione tra due sprite. Per far sì che la chiamata sia effettivamente attiva è necessaria un'istruzione SPRITE ON

SQU(Y)

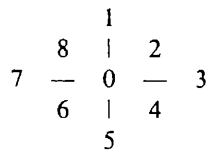
Ritorna la radice quadrata dell'espressione Y che deve essere maggiore o uguale a zero.

STICK(Y)

Y può assumere valori da 0 a 2:

0 Cursore
1 Joystick porta 1
2 Joystick porta 2

Viene ritornata la direzione secondo il seguente schema di trascodifica:

```
        1
    8   |   2
7   —   0   —   3
    6   |   4
        5
```

STOP

Istruzione che provoca l'interruzione dell'esecuzione del programma; nello stesso tempo viene visualizzato il messaggio:

"Break in"

numero di linea.

STOP ON/ OFF / STOP

Attiva, disattiva o sospende la chiamata a un sottoprogramma particolare allorché vengono premuti i tasti di CTRL e STOP. Per maggiori dettagli si consulti il Capitolo 2.

STRIG(Y)

L'espressione intera Y deve essere compresa tra 0 e 4:

0 Barra di spaziatura
1 Segnale 1 dal joystick 1
2 Segnale 1 dal joystick 2
3 Segnale 2 dal joystick 1
4 Segnale 2 dal joystick 2

Quando si preme il tasto del joystick viene ritornato —1, 0 altrimenti.

STRIG (Y) ON / OFF / STOP

Attiva, disattiva o sospende la rilevazione di quando viene premuto un particolare tasto di joystick (o la barra spaziatrice). Y, come nel caso precedente, deve essere tra 0 e 4. Per definire il sottoprogramma che viene attivato alla rilevazione dell'evento di cui sopra, è necessario usare un'istruzione ON STRIG GOSUB.
Esempio: STRIG(0) ON abilita la rilevazione della pressione della barra di spaziatura.

STRING$(x,Y)

Variante: STRING$(x,Y$)

Ritorna una stringa di lunghezza x contenente solo i caratteri che hanno codice di carattere Y, oppure il primo carattere di Y$. Se il primo caratere è grafico, viene ritornata una stringa di CHR$(1).
Esempio: STRING$(4,65) ritorna "AAAA".

STR$(Y)

Ritorna la rappresentazione in formato stringa del valore numerico dell'espressione Y. In questo modo si dispone di un conveniente metodo di conversione da esadecimale, ottale o formato esponenziale.

SWAP Y,X

Scambia i valori delle variabili indicate. Entrambe devono essere dello stesso tipo.
Esempio: A$="w":B$="e":SWAP A$,B$:? A$ visualizzerà "e".

TAB(Y)

Funzione che può essere usata solo dopo le istruzioni PRINT o LPRINT. TAB sposta la stampa della prossima entità alla colonna Y - a meno che vi si trovi già, nel qual caso non viene intrapresa alcun'azione. Y deve essere compreso tra 0 e 255.
Esempio:PRINT TAB(4);"Sulla quarta colonna"

TAN(Y)

Funzione trigonometrica che ritorna il valore della tangente dell'espressione Y, valutata in radianti.

## TIME

Speciale variabile intera che viene azzerata al momento dell'accensione e incrementata ogni 1/50 di secondo. Questo aggiornamento avviene durante l'interruzione di sistema del VDP che viene però sospesa durante le operazioni di ingresso e uscita su nastro.

## TRON / TROFF

L'istruzione TRON fa in modo che ogni linea del programma venga visualizzata non appena è stata eseguita. Quest'istruzione può essere attivata sia in modalità diretta sia in quella indiretta (interpretata) e viene negata dalla TROFF.
Esempio:
10 TRON
20 REM
30 REM
40 TROFF
provocherà la visualizzazione di [20],[30],[40]

## Z=USR (Y)

Variante: Z=USR x (Y)

Quest'istruzione esegue una chiamata alla routine in codice macchina posta alla locazione specificata dall'istruzione DEFUSRx. Il valore dell'espressione Y viene passato alla routine. Se non deve essere trasferito alcun parametro è necessario dare un valore fittizio a Y.
Al completamento della routine in codice macchina viene restituito un valore che viene assegnato a Z.
Sia al momento della chiamata che in quello del ritorno, l'elemento scambiato si trova alla locazione &HF7F6. Il tipo di dato è indicato dal valore dell'accumulatore:

*Intero:* 2
*Stringa:* 3
*Singola precisione:* 4
*Doppia precisione:* 8

La locazione di un elemento stringa viene comunicata per mezzo della coppia DE di registri. Un tal elemento consiste di tre byte:

1. Il numero di caratteri della stringa.
2. e 3. La locazione effettiva ove è memorizzata la stringa.
Esempio:
10 DEFUSR4=&HD000
20 Y=USR 4 (X)

VAL(Y$)

Ritorna il valore numerico della stringa Y$. Qualsiasi spazio, carattere di tabulazione o linea nuova viene ignorato.

VARPTR(Y)

Variante: VARPTR(# numero di file)

Funzione che ritorna la prima locazione della variabile specificata.
È applicabile a un qualsiasi tipo di variabile. Se viene ritornato un indirizzo negativo, è necessario aggiungervi 65536 per ottenere la locazione reale desiderata.
Per maggiori dettagli si consulti la sezione 'Memorizzazione del programma' al termine del Capitolo 2.
La variante con il numero di file consente di ottenere l'indirizzo del primo elemento del blocco di controllo del file.

VDP(Y)

Si tratta di una speciale variabile in cui Y deve essere compreso tra 0 e 8. I valori da 0 a 7 ritornano il valore del registro a sola scrittura del VDP. VDP(8) permette di ottenere il valore del registro di stato del VDP. Per maggiori dettagli si faccia riferimento alla sezione sul VDP.

VPEEK(indirizzo)

Ritorna il valore contenuto nell'elemento della VRAM di cui si è specificato l'indirizzo. Quest'ultimo deve essere compreso tra 0 e 16383.

VPOKE X,Y

Funzione per scrivere il valore Y all'indirizzo X nella VRAM. Y deve essere compreso nell'intervallo tra 0 e 255.

### WAIT porta, valore di un byte

Variante: WAIT porta, valore di un byte, maschera.

Istruzione che sospende il programma in attesa che un byte venga acquisito dalla porta indicata. La configurazione dei bit del byte acquisito deve essere la medesima del valore del byte specificato nell'istruzione.
La variante consente invece di operare dapprima un OR tra la maschera e il byte in ingresso e successivamente di effettuare un AND tra i bit del risultato e quelli del byte specificato.
Tutte le suddette espressioni devono assumere valori interi tra 0 e 255.

### WIDTH larghezza dello schermo di testo

Istruzione che consente di precisare la larghezza dello schermo in entrambe le modalità testo.
I valori indicati devono essere compresi:

a) tra 1 e 40 per la modalità testo 40 * 24.
b) tra 1 e 32 nella modalità 32 * 24.

Esempio: WIDTH 20.

Capitolo 4

# IL LINGUAGGIO MACCHINA DELLO Z-80

## Microprocessori. L'architettura Z-80. Le istruzioni. I modi di indirizzamento

### Microprocessori

Il cuore di ogni elaboratore basato sul sistema MSX è costituito da un microprocessore Z-80. Esso può essere programmato direttamente usando il codice macchina e possiede tre caratteristiche essenziali:

1. Semplicità nello svolgimento di operazioni aritmetiche.
2. Spostamento di dati tra locazioni diverse.
3. Operazioni logiche.

Un programma, chiamato interprete BASIC, è scritto in codice macchina nella ROM di sistema (read only memory = memoria a sola lettura). Questo programma traduce le istruzioni BASIC in codice macchina che quindi vengono eseguite dallo Z-80.
La mancanza di questa fase di traduzione consente ad un programma scritto in codice macchina di poter essere eseguito più rapidamente rispetto ad uno analogo scritto in BASIC. Da qui deriva la definizione del BASIC che lo indica come linguaggio ad "alto livello" che permette l'uso di comandi non direttamente disponibili al livello intrinseco del microprocessore.
Quindi la velocità è il principale vantaggio che si ottiene dall'uso del codice macchina, ma quali sono gli svantaggi? Così come il BASIC è stato progettato per consentire l'utilizzo e la programmazione dell'elaboratore con poca esperienza pratica, l'uso del linguaggio macchina richiede un più alto grado di conoscenza di quel che accade all'interno della macchina, dei sistemi di numerazione binario ed esadecimale - si tratta comunque di piccoli inconvenienti rispetto ai sorprendenti risultati che si possono ottenere.

Sappiamo già che nei sistemi che stiamo considerando c'è uno Z-80, affiancato da un 9129 VDP (processore video) e da un integrato per la produzione di suoni, l'AY-3-8910; ma come sono interconnessi tra loro?
La comunicazione tra le varie unità avviene attraverso due principali canali: il bus dati ed il bus indirizzi. Il bus indirizzi è costituito da 16 linee che corrono dallo Z-80 fino ai blocchi di memoria RAM e ROM. Il bus dati, parallelo al primo, è composto solo da otto linee.
Usando le 16 linee del bus indirizzi, lo Z-80 è in grado di accedere ad una qualsiasi locazione tra 0 e 65535. Ciascuna locazione, sia essa in memoria RAM o ROM, può contenere un valore tra 0 e 255.
Quando lo Z-80 scrive un indirizzo sul bus indirizzi e indica alla memoria che deve essere effettutata un'operazione di lettura piuttosto che una di scrittura, il numero contenuto alla locazione indicata viene automaticamente messo sul bus dati. Quindi il processore lo copia in un registro interno.
Poiché il sistema necessita di essere esattamente sincronizzato viene usato, come riferimento, un "temporizzatore elettronico" (clock), esterno allo Z-80, che lavora ad una frequenza di 3.57 MHz.
Anche il VDP ha accesso al bus dati per consentire alla CPU (unità centrale di elaborazione, lo Z-80) di passare dati ed istruzioni. Inoltre il VDP possiede una RAM di 16 Kbyte, alla quale solo lui può accedere direttamente.
Per questo motivo esso dispone di un bus ad 8 bit 'privato' e bidirezionale.
A partire dalle istruzioni d'impostazione della modalità di visualizzazione e dal contenuto della VRAM, il VDP costruisce la schermata che deve essere visualizzata. Quest'ultima viene trasmessa direttamente a un monitor o viceversa modulata per un apparecchio TV domestico.
Al momento dell'accensione, la definizione dell'insieme dei caratteri conservata nella ROM viene trasferita dalla CPU (lo Z-80) al VDP e da questo alla VRAM.

## Organizzazione del sistema

La gestione di aree di memoria di dimensioni pari a quelle suddette viene semplificata se si considera l'intera area di cui si dispone come divisa logicamente in blocchi dalle dimensioni più piccole. Una divisione piuttosto comune è quella che fa riferimento a pagine di 256 locazioni ciascuna. La prima di queste pagine è la pagina 0 e contiene gli indirizzi assoluti dallo 0 a 255. Il termine "pagina" può anche riferirsi a un blocco di memoria di 16K byte. Il tempo impiegato dalla CPU per trasferire dati da e sulla pagina zero è inferiore a quello richiesto nel caso in cui si debba attraversare il confine tra due pagine. Di conseguenza la maggior parte dei piccoli elaboratori riservano la pagina zero ad uso del sistema.

Le istruzioni per il microprocessore sono in forma di uno o due byte. Se inoltre è necessario specificare, come operandi dell'istruzione, dati o indirizzi, devono essere usati altri byte.
Ne consegue che un programma in linguaggio macchina ha la forma di una sequenza continua di byte di istruzioni e dati.
Certe locazioni di memoria interne al processore sono accessibili al programmatore. Le più utili tra queste sono:

1. Il "program counter" (contatore che memorizza l'indirizzo della prossima istruzione da eseguire).
2. Il puntatore dello stack (o pila).
3. Due registri indice: IY e IX.
4. L'accumulatore, o registro A.
5. Sei registri generali da B a L.
6. Il registro di "flag" F.
7. I registri d'interruzione e di refresh I ed R.
8. Un insieme alternativo di registri da A' a L'.

I registri dall'1 al 3 dispongono di 16 bit e possono quindi contenere un qualsiasi numero tra 0 e 65535. I restanti sono da otto bit e sono limitati all'intervallo da 0 a 255.
Poiché il processore deve sapere dove sono poste le istruzioni, il program counter assolve a questa funzione mantenendo sempre memoria dell'indirizzo della prossima istruzione che deve essere eseguita.
Il registro di "flag" mantiene informazioni che concernono sia lo stato del sistema, sia il risultato di alcune determinate operazioni. Per esempio, se il risultato di una sottrazione fosse negativo, verrebbe posto a 1 il bit di segno (S) del registro F.
Il registro di flag viene usato da quelle istruzioni che consentono di effettuare salti condizionati all'interno del programma. Queste istruzioni fanno sì che il program counter punti alla locazione dell'istruzione voluta se uno dei bit del registro F vale 1 o piuttosto 0.
I due registri indice IY ed IX possono essere caricati con qualsiasi valore, compreso nell'intervallo di validità, desiderato dal programmatore. Principalmente questi indirizzi sono stati concepiti per mantenere indirizzi di dati o di procedure.
I sei registri generali: B, C, D, E, H, L possono anch'essi contenere un qualsiasi valore; in alternativa possono essere usati a coppie, BC, DE, HL, per fornire registri a 16 bit.
Il registro A o accumulatore ha una particolare importanza che gli deriva dall'architettura dello Z-80. La maggior parte delle operazioni logiche ed aritmetiche su 8 bit richiedono che uno dei dati di partenza sia stato preventivamente caricato nell'accumulatore. Dopo l'operazione, il risultato viene nuovamente posto in A.

Sebbene questo consenta di eseguire le suddette operazioni in modo rapido, vengono in tal modo a rendersi necessarie istruzioni per il caricamento dell'accumulatore e per il trasferimento finale del risultato. Lo Z-80 può anche effettuare addizioni o sottrazioni a 16 bit. In questo caso la coppia di registri HL sostituisce il registro A. L'insieme dei registri da A' a L' può essere selezionato alternativamente all'insieme da A a L: in questo modo si dispone di due insiemi separati per la memorizzazione e gestione dei dati. In pratica però il secondo insieme è usato assai raramente - usualmente per consentire un rapido servizio delle procedure di gestione delle interruzioni.

## Rappresentazione binaria ed esadecimale

Prima di affrontare questa sezione, provate a rispondere a questi due quesiti:

1. Quali sono le rappresentazioni binarie di —34 e di 0.02?
2. Qual è il numero decimale equivalente all'esadecimale &H0FDE ?

Poiché i microprocessori non sono stati progettati per usare il sistema di numerazione decimale, chi programma in codice macchina deve essere a perfetta conoscenza sia del sistema di numerazione binario sia di quello esadecimale.
In decimale si possono usare dieci cifre, in binario soltanto due: 0 e 1, quindi un numero binario è una serie di 0 e di 1.
Un'altra importante differenza è la seguente: in decimale il passaggio da una posizione all'altra nella scrittura di un numero si ha quando si passa una decina mentre in binario la si ha per potenze di due.
Per convertire un numero binario in decimale, è necessario sommare i valori rappresentati nelle colonne ove c'è un 1. Nella figura 4.1 l'equivalente in binario di 100 viene calcolato come somma di potenze di due, cioè come 64 + 32 + 4. Ciascun uno o zero viene rappresentato da un bit, mentre una serie di otto bit viene detta byte.
Chiaramente il massimo numero rappresentabile in un byte è 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255. Per numeri più piccoli è comunque conveniente indicare gli zeri che sono nei primi degli otto bit.
Il bit più a destra è detto meno significativo (l.s.b. dall'inglese least significant bit) mentre quello più a sinistra è il bit più significativo (m.s.b. dall'inglese most significant bit). Queste abbreviazioni non vanno comunque confuse con L.S.B. (byte meno significativo) e M.S.B. (byte più significativo).
Per esempio, in un indirizzo a sedici bit:

1111111100000010

| Decimale | Rappresentazione Binaria | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 = 8*1 + 4*0 + 2*0 + 1*1 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 100 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 200 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 = 128 + 64 + 8 |

*Fig. 4.1 - Valori binari*

il l.s.b. è 0, il m.s.b. è 1 mentre il M.S.B. è 1111111.
Nei capitoli seguenti avremo spesso bisogno di riferirci ad un bit particolare nell'ambito di un byte. Useremo la notazione convenzionale:

7 6 5 4 3 2 1 0

L'addizione e la sottrazione in binario sono piuttosto semplici con una eccezione - la rappresentazione di un numero negativo. Poiché non si dispone di qualcosa di analogo al segno meno, i numeri negativi sono rappresentati diversamente. Il metodo per codificarli è la notazione in complemento a due. Questa si ottiene considerando il valore assoluto del numero che si vuole convertire, rappresentandolo in binario, invertendo tutti i bit ed infine aggiungendo 1.
Per esempio, per ottenere il complemento a due di —12 considerate la rappresentazione binaria del suo valore assoluto (12) che è 00001100. Invertendo i bit si ottiene 11110011 ed aggiungendo 1 si ha 11110100. Poiché il processore deve poter distinguere questo numero da 244, che ha la medesima rappresentazione, vengono imposti i limiti superiore ed inferiore ai valori che possono essere rappresentati in un byte: —128 e 127. Questo evita ogni sovrapposizione.

| Operazione | Esempio |
|---|---|
| Valore negativo | —121 |
| Valore assoluto | 121/01111001 |
| Forma inversa | 10000110 |
| Incremento | 00000001 |
| Complemento a due | 10000111 |

*Fig. 4.2 - Rappresentazione di numeri negativi in binario*

La divisione per due di un numero binario si ottiene muovendo tutti i bit di una posizione verso destra. Quindi per ottenere la rappresentazione binaria di 1/2, il byte 00000001 viene spostato quindi da 0.1. La figura 4.3 mostra i valori binari frazionari.

| Decimale | Frazione binaria |
|---|---|
| 0.5 | 0.1 |
| 0.25 | 0.01 |
| 0.125 | 0.001 |
| 0.0625 | 0.0001 |
| 0.03125 | 0.00001 |

*Fig. 4.3 - Rappresentazione di frazioni in binario*

Ogni numero rappresentato su otto bit può essere spezzato per dar luogo a due semi-byte ("nibble"). Ciascuno di questi può rappresentare, isolatamente, un numero da 0 a 15. In questo modo si ottiene un utile e abbreviata rappresentazione di dati binari: occorrono chiaramente 16 differenti cifre, ragion per cui, oltre alle solite da 0 a 9, si usano le prime lettere dell'alfabeto, da A a F.

| **Decimale** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Esadecimale** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 |

*Fig. 4.4 - Notazione esadecimale*

Questo metodo di rappresentazione è il sistema di numerazione esadecimale. Per esempio, per convertire 56 decimale in esadecimale si consideri dapprima la rappresentazione binaria: 00111000 da cui si estrae il valore

dei due semi-byte: 0011 e 1000, vale a dire 3 e 8; se ne deduce che la forma esadecimale di 56 è 38.
Ciò che segue dimostra un'altra semplificazione rispetto al sistema binario. Per convertire un numero di due cifre esadecimali in decimale, è sufficiente moltiplicare per 16 la cifra più a sinistra e poi aggiungervi quella sulla destra.
Come avrete intuito la base del sistema esadecimale è 16. Quindi per convertire da decimale a esadecimale è solo necessario scomporre il valore secondo potenze di 16.

| | | | | | |
|---|---|---|---|---|---|
| **Esadecimale** | 4 | 3 | A | 4 | |
| **Valore posizionale** | 4096 | 256 | 16 | 1 | |
| **Valutazione** | 4*4096+ | 3*256+ | 10*16 | +4 | =17316 |

*Fig. 4.5 - Conversione da esadecimale a decimale*

## Operazioni logiche

Considerate il seguente problema:

Dati due numeri binari di 8 bit, si richiede di formare un terzo numero binario in cui ogni bit sia a 1 se, e soltanto se, i bit omologhi di entrambi i numeri dati sono a 1.
La risposta? Semplice, prima caricate uno dei due valori dati nel registro A, quindi effettuate l'operazione logica di AND tra esso e l'altro byte. Al termine dell'operazione il risultato viene ritornato nel registro A.
L'operazione di AND è una delle tre operazioni logiche che lo Z-80 è in grado di compiere: i bit di due byte vengono confrontati a uno a uno e, in base al risultato del confronto, un nuovo valore viene scritto nell'accumulatore. Come abbiamo visto l'operazione di AND mette a 0 tutti i bit tranne quelli in cui entrambi gli operandi hanno 1. L'operazione inversa a questa è l'istruzione OR, che ritorna 1 in ciascun bit dell'accumulatore quando uno o entrambi i bit degli operandi nella stessa posizione sono a 1. La terza istruzione, XOR, è un incrocio delle due precedenti. Perché in un bit dell'accumulatore sia posto 1 è necessario che i bit di analoga posizione negli operandi siano tra loro diversi.
A chi affronta per la prima volta i problemi del codice macchina queste operazioni sembreranno senza importanza: al contrario esse sono spesso necessarie per certi calcoli e per il controllo di particolari funzioni.

| Operazione | AND | XOR | OR |
|---|---|---|---|
| Condizione per risultato uguale a 1 | Entrambi gli operandi a 1 nella colonna | I bit dei due operandi sono diversi | Uno o entrambi gli operandi sono a 1 |

*Fig. 4.6 - Confronto tra le operazioni logiche di base*

## L'architettura Z-80

La Zilog ha realizzato molte versioni dello Z-80 che si differenziano per la massima velocità a cui possono funzionare. Per lo Z-80A, utilizzato nel MSX, la massima velocità di sincronismo è di 4 Mhz (mentre lo Z-80B può arrivare a 6MHz).
La configurazione strutturale del processore è un tipico 40 piedini "Dual in Line Package" (D.I.L.P.). Il bus indirizzi richiede 16 linee e il bus dati otto. I principali piedini rimanenti hanno le seguenti funzioni:

1. Alimentazione
2. Piedini di reset e di interruzione: RESET, INT e NMI.
3. WAIT: impone una pausa al processore, tipicamente per consentire una ricerca in memoria dei dati necessari.
4. BUSREQ e BUSAK - linee di controllo che consentono ad altri processori di accedere ai bus dati ed indirizzi.
5. RFSH - indica che le 7 linee di indirizzo più basso sono in uso per il refresh di parte della memoria.
6. Piedini di ingresso per il sincronismo di sistema (clock).
7. Piedini di lettura e scrittura (RD e WR).
8. Piedini MREQ e IORQ.

Inoltre, per poter trasferire dati da e sulla memoria, il processore può acquisire o scrivere dati su porte a 8 bit, in numero massimo di 256. I principali registri di controllo del sistema sono accessibili in questo modo. Questo consente di iniziare la maggior parte delle operazioni di ingresso/uscita scrivendo semplicemente sulla porta interessata.
Per ottenere l'attenzione del processore onde poter compiere una funzione urgente, un componente del sistema può generare un'interruzione. In questo caso lo Z-80 finisce di eseguire l'istruzione corrente e poi salta ad eseguire la procedura di gestione dell'interruzione. Dopo il suo completamento, il controllo ritorna al programma che era in esecuzione in precedenza. Spesso non è necessario ricorrere ad una procedura d'interruzione qualora

sia richiesto solo l'uso del bus dati o indirizzi. Infatti lo Z-80 è in grado di isolarsi dai due bus. La linea BUSREQ viene usata per richiedere all'unità centrale l'uso dei bus, e quest'ultima usa la linea BUSAK per notificare il suo avvenuto isolamento.
La linea MREQ viene usata per indicare che il processore deve effettuare una lettura o una scrittura in memoria e quindi ha messo un indirizzo sul bus indirizzi. La linea IORQ viene abbassata sia per comunicare che una interruzione è stata ricevuta, sia per indicare che sta avendo luogo un'operazione di I/O.
Internamente lo Z-80 comprende un'unità aritmetico logico (A.L.U.), una unità di controllo, alcuni registri e della memoria a sola lettura. Il loro modo di funzionamento viene illustrato chiaramente se si considera un esempio:

**i.** Viene fatto un accesso all'indirizzo puntato dal contatore di istruzioni del programma (program counter) e il suo contenuto viene caricato nel registro istruzioni. Nel nostro esempio sia tale valore 01111100, 124 in decimale. Quest'istruzione viene interpretata dal processore come un ordine di caricare il contenuto del registro H nell'accumulatore A.

**ii.** L'unita di controllo centrale incrementa il contatore di istruzioni del programma che così punta ora all'istruzione successiva.

**iii.** Il byte che si trova nel registro H viene copiato nel registro A. In tal modo ogni suo precedente valore viene sovrascritto.

**iv.** Nel processore viene caricata l'istruzione successiva.

L'intera operazione viene completata in 4 T cicli (periodi del sincronizzatore esterno). Questo dà un'idea della temporizzazione del sistema - e chiarisce inoltre perché gli errori di programma siano così difficilmente rintracciabili! Sebbene in confronto a molti altri processori a 8 bit lo Z-80 sia ben fornito di registri interni, essi spesso si rivelano insufficienti per memorizzare le informazioni di cui abbiamo bisogno sottomano. Fortunatamente lo Z-80 dispone di uno spazio di memoria più ampio, lo stack o pila.
Lo stack è una sezione di memoria che può essere usata per memorizzare dati di due byte ciascuno. Tali dati possono provenire da uno qualsiasi dei registri a 16 o bit da una coppia di altri registri, ad eccezione del contatore d'istruzioni del programma.
Il nome stack, pila, è appropriato, solo che in realtà, in memoria la pila è rovesciata! Infatti quando una coppia di byte viene aggiunta, o messa sullo stack, la sua cima viene spostata due posizioni più in basso nella memoria. La locazione dell'ultimo byte della coppia messa sullo stack viene ricordata dal registro puntatore allo stack.
Uno svantaggio implicito nell'uso dello stack è che le coppie di byte devono essere usate sulla base di una strategia "ultimo entrato primo uscito" (in inglese L.I.F.O. = last in first out).

*Esempio:* Il contenuto del registro a 16 bit IY viene messo sullo stack seguito dal contenuto della coppia di registri BC. Per accedere alla prima coppia messa sullo stack è necessario prima utilizzare la seconda. Lo stack viene inoltre usato per memorizzare gli indirizzi di ritorno da sottoprogrammi o da procedure di gestione d'interruzioni.
Prima che vi sia presentato l'insieme delle istruzioni macchina, alcune parole sulle 256 porte di ingresso/uscita. È possibile leggere o scrivere un dato di un solo byte da una qualsiasi porta usando l'istruzione appropriata. Il processore usa le otto linee meno significative, da A0 a A7, per selezionare la porta (non viene fatta una lettura o scrittura in memoria poiché è alta la linea IORQ e non la MREQ). Il dato viene quindi posto sul bus dati e viene utilizzato o dallo Z-80 (READ) o dalla porta (WRITE).

## Le istruzioni dello Z-80

Lo Z-80 ha 158 istruzioni macchina che la Zilog raggruppa in 11 categorie. Prima di considerarle, è necessario comprendere in che modo scrivere o acquisire programmi in linguaggio macchina.
Dando un rapido sguardo alle pubblicazioni sui computer ci si rende conto che esistono due metodi comuni per rappresentare le procedure scritte in linguaggio macchina. Il più diffuso di questi metodi consiste nel listare i valori esadecimali che, rappresentandone le istruzioni, compongono il programma stesso. L'utente scrive semplicemente questi valori nelle appropriate locazioni di memoria: il programma viene poi eseguito accedendo alla prima locazione ad esso riservata, per mezzo dell'istruzione USR del BASIC. I byte in questione vengono detti codice oggetto. in questo formato la sequenza delle istruzioni può essere elaborata direttamente dallo Z-80.
Però mettere manualmente in memoria un programma byte dopo byte è un procedimento laborioso e con un alto rischio di errore. Una semplice soluzione consiste nello scrivere una routine BASIC che acquisisca numeri esadecimali di due cifre. Essa stessa metterà poi in memoria, in una serie di locazioni, l'equivalente decimale di tali numeri. Un programma BASIC di questo genere viene di solito detto un "caricatore esadecimale"; di seguito ne viene fornito un esempio:

```
                    CARICATORE ESADECIMALE
200 SCREEN 1:COLOR 10,1,1:CLS
210 LOCATE 2,2:?"HEX LOADER FI TO FINISH":?
220 INPUT"START LOCATION";ST
230 SW=ST—1:?
```

```
240 SW=SW+1:?SW;" ";HEX$(SW);"??";
250 ?CHR$(29);CHR$(29);
260 A$=INPUT$(2)
270 IF A$="FI" OR A$="fi" THEN 320
280 A$(1)=MID$(A$,1,1):A$(2)=MID$(A$,2,1)
290 IF INSTR("0123456789ABCDEFabcdef",A$(1))=0 OR INSTR
    ("0123456789ABCDEFabcdef",A$(2))=0 THEN 260 ELSE ?A$;
300 A=VAL("&H"+A$):?" ";A
310 POKE SW,A:?:GOTO 240
320 ?:?:?:?"START ";ST;" END ";SW—1
```

Ciò nonostante, se il programma in linguaggio macchina è di una lunghezza non indifferente, il procedimento di cercare l'equivalente in esadecimale di ogni istruzione diventa di scarsa praticità e quindi è necessario usare un assemblatore.
Un programma assemblatore (o Assembler, una sorta di compilatore per il linguaggio Assembly) consente di scrivere una procedura usando dei simboli mnemonici per le istruzioni e, se necessario, stringhe per indicare indirizzi. Il file sorgente viene tradotto in codice oggetto per mezzo dell'assemblatore. Quando si è raggiunta una certa familiarità con le istruzioni simboliche, la scrittura di programmi in codice macchina risulta agevole tanto quanto scriverli usando il BASIC. Il file sorgente di solito consiste di linee numerate, ciascuna delle quali contiene una o più istruzioni in linguaggio macchina. Tutti gli assemblatori che si rispettino permettono l'uso dell'editor a schermo intero ed hanno comandi di una sola lettera.
La maggior parte degli assemblatori reperibili in commercio consiste di tre pacchetti:

1. Assemblatore
2. Monitor
3. Disassemblatore

Una volta che il file sorgente è completo, esso viene assemblato e se ne ottiene un codice oggetto. Il file viene quindi salvato poiché potrebbe contenere degli errori per i quali sia necessario spegnere l'elaboratore prima di poter rieseguire il programma. Alla fine si fa uso del monitor per seguire l'esecuzione - una istruzione alla volta se necessario. Il disassemblatore viene usato per esaminare e cambiare aree di memoria e di solito fa ad esso riferimento il monitor.
Oltre a queste operazioni di base viene fornito un insieme secondario di istruzioni tipicamente per spostare, salvare o caricare blocchi di memoria, effettuare operazioni di ricerca su stringhe, ecc.

Secondo l'attività di programmazione che si sta svolgendo al momento, si può scegliere quale dei tre suddetti programmi utilizzare. Il prezzo di questi pacchetti è accessibile: da qualche migliaio di lire in sù. Per un uso saltuario può essere sufficiente un sistema basato su nastro a cassetta, ma per un impiego più serio è quasi obbligatorio ricorrere a periferiche più veloci. La ragione principale di ciò è che quasi tutti i programmi in linguaggio macchina conteranno inizialmente degli errori, e dover ricaricare l'intero pacchetto dopo ogni errore, in seguito al quale la situazione non è ripristinabile, può diventare causa di irritazione.
Sebbene esistano 158 tipi di istruzioni supportati dal processore, i codici operativi singoli sono in totale 666. Questo accade perché la maggior parte delle categorie di istruzioni contiene un insieme di operazioni, ciascuna delle quali compie la medesima operazione delle altre, ma consente di indicare gli indirizzi in modo diverso.
Lo Z-80 consente di usare 10 diverse modalità di indirizzamento, modalità che verranno esaminate più dettagliamente nella prossima sezione. Il menù delle istruzioni di cui si dispone può essere diviso nelle seguenti categorie:

1. Istruzioni di caricamento a 8 e 16 bit.
2. Istruzioni aritmetiche a 8 e 16 bit.
3. Istruzioni logiche a 8 bit.
4. Istruzioni di rotazione e spostamento laterale di bit.
5. Istruzioni sui bit.
6. Istruzioni di salto e di sottoprogramma.
7. Istruzioni di trasferimento di blocchi e di ricerca.
8. Istruzioni di controllo dell'unità centrale e dell'I/O.

## Istruzioni di caricamento a 8 e 16 bit

Un'istruzione di caricamento a 8 bit copia il contenuto di un registro o di una locazione di memoria in un altro registro o locazione di memoria. L'istruzione a 16 bit effettua un'operazione equivalente tra un registro a 16 bit e la memoria o un altro registro.
Si noti che quando un valore a 16 bit viene memorizzato alla locazione X, il byte più significativo viene memorizzato nella locazione X + 1, mentre quello meno significativo finisce nella locazione X. In modo analogo se viene caricato un registro a 16 bit con due byte alla locazione X, allora il valore contenuto all'indirizzo X + 1 viene caricato negli 8 bit più significativi del registro, mentre il contenuto della locazione X finisce negli 8 bit meno significativi.
L'istruzione simbolica per l'assemblatore Zilog per questa operazione è LD X,Y dove Y è il byte che vien copiato nella locazione o registro X.

*Esempi:*

LD A,B carica nel registro A il contenuto del registro B.
LD H,A carica nel registro H il contenuto del registro A.
LD B,40 carica 40 nel registro B.
Se un operando è racchiuso tra parentesi, indica l'indirizzo del dato che deve essere usato.
LD A,(40) carica nell'accumulatore il contenuto della locazione 40.

In modo analogo LD B,(HL) carica nel registro B il byte il cui indirizzo è nella coppia HL di registri. Ecco infine due esempi di caricamento a 16 bit:

LD HL,(40) carica nel registro L il contenuto della cella 40 e in H quello della cella 41.
LD IY,22 carica 22 nel registro indice IY.

## Istruzioni aritmetiche a 8 e 16 bit

Le più semplici sono le istruzioni per incrementare e per decrementare: servono per aggiungere o sottrarre 1 al dato indicato. In simbolico sono INC e DEC.
*Esempi:*

INC A aggiunge 1 al valore dell'accumulatore.
DEC IX sottrae 1 al valore del registro indice IX.

Esistono due istruzioni simboliche per ognuna delle due operazioni di addizione e sottrazione: ADD e ADC per l'addizione e SUB e SBC per la sottrazione. ADD e SUB hanno l'ovvio significato che ci si attende da queste operazioni. In particolare per operazioni a 8 bit il risultato viene posto nel registro A, mentre in quelle a 16 bit il risultato va nella coppia di registri HL. Per sommare 4 e 5, prima si carica A con uno dei due valori:

```
LD A,4
```

e quindi si aggiunge 5:

```
ADD A,5
```

Il risultato sarà contenuto nel registro A. Come abbiamo detto per le addizioni e sottrazioni a 8 bit è obbligatorio usare il registro A. Analogamente, nelle operazioni a 16 bit, si deve dapprima caricare uno dei valori nella coppia HL e in essa verrà posto il risultato (con l'eccezione, che conferma la rego-

la, che un registro indice può essere usato come accumulatore per una certa categoria di addizioni a 16 bit).
*Esempio:* ADD HL,BC somma il contenuto della coppia BC al valore presente nella coppia HL. Il risultato è in HL.
Le istruzioni ADC e SBC significano somma con riporto e sottrazione con riporto (dall'inglese Add e Subtract with Carry); in realtà il riporto viene memorizzato in uno dei bit del registro flag F. Questo bit viene posto a 1 se un'addizione ha dato un risultato troppo grande per essere caricato nell'accumulatore oppure se in una sottrazione è necessario l'uso del riporto.
Quando viene eseguita una ADC, il dato specificato più il riporto vengono sommati all'accumulatore. Per esempio ADC A,4 sommerebbe 5 al valore contenuto nell'accumulatore, se il flag di riporto fosse a 1.
Sono disponibili anche istruzioni per agire direttamente sul flag di riporto:

1. CCF: il valore del bit di riporto viene invertito.
2. SCF: il bit di riporto viene posto a 1.

Si noti che non esiste un'operazione di sottrazione per dati su 16 bit. Ne consegue che non è necessario indicare esplicitamente il registro A quando si usa l'istruzione SUB.
*Esempio:* SUB 4 decrementa l'accumulatore di 4.

## Istruzioni logiche a 8 bit

Tutti i sei tipi di operazioni in oggetto richiedono che uno dei dati sia stato preventivamente caricato nell'accumulatore. Il risultato è ritornato nell'accumulatore.
Tre operazioni di questa categoria sono state presentate in precedenza:

1. AND
2. OR
3. XOR

Le altre tre sono:

1. CP per confrontare
2. CPL per ottenere il complemento a 1.
3. NEG per ottenere il complemento a 2.

L'istruzione CPL inverte semplicemente il valore di ogni bit nell'accumulatore. Anche NEG inverte i bit, ma incrementa il risultato per ottenere il complemento a due del valore originale.

L'istruzione CP è invece insolita in quanto il dato viene sottratto all'accumulatore, ma il risultato non viene ritornato; il valore dell'accumualtore rimane invariato: in compenso viene cambiato il valore di parecchi bit del registro flag F; in particolare:

1. Il bit denominato ZERO viene posto a 1 se il valore dell'accumulatore è uguale al dato confrontato ad esso con la CP. Per esempio CP 4 porrà a 1 il flag Z (ZERO) se nell'accumulatore c'è 4, altrimenti Z viene posto a 0.
2. Il bit di segno (S) viene posto a 1 se il dato confrontato è maggiore di quello contenuto nell'accumulatore. Inoltre viene posto a 1 il flag N, mentre sono aggiornati i flag H, P/V e CY.

*Esempi:*

AND 128 opererà un'operazione di prodotto logico tra 10000000 (binario) e l'accumulatore A.
CPL inverte i bit dell'accumulatore.

## Istruzioni di rotazione e spostamento laterale di bit

Se i bit di un numero binario vengono spostati lateralmente di una posizione sulla sinistra, il valore viene raddoppiato. Quindi lo spostamento laterale dei bit che compongono un byte è uno dei modi con cui il processore è in grado di svolgere moltiplicazioni e divisioni.
Lo Z-80 dispone di tre operazioni di spostamento e di otto per la rotazione di bit, ciascuna delle quali opera solo su dati di otto bit. Quando i bit di un byte vengono spostati lateralmente in una direzione, un bit viene perso e un altro viene lasciato vacante. Il bit di riporto viene usato per sostituire o per memorizzare il bit vacante o quello perso in dipendenza dell'istruzione usata.
Le otto operazioni di rotazione consistono in quattro istruzioni che agiscono sull'accumulatore, e di altre quattro, funzionalmente analoghe, che agiscono su una qualsiasi locazione di memoria o su un differente registro:

1. **RL e RLA:** hanno la stessa funzione, solo che la seconda è un'istruzione dedicata all'accumulatore. Ogni bit viene spostato di una posizione a sinistra. Il bit vacante, quello meno significativo, viene riempito con il bit di riporto, mentre quello più significativo, che andrebbe perso, viene memorizzato nel bit di riporto stesso.

*Esempi:* RL B
RL (HL)
RLA

**2. RR e RRA:** analoghe a RL e RLA, tranne che la rotazione viene effettuata verso destra.
*Esempi:* RRH
RR (HL)
RRA

**3. RLC e RLCA:** in questo caso il bit più significativo viene posto sia nel bit di riporto sia nel bit meno significativo, lasciato vacante:
*Esempi:* RLC (HL)
RLC A
RLCA

Si noti che sia RLC A sia RLCA effettuano la medesima operazione. Però, come gran parte delle operazioni, provocano delle modifiche in molti dei bit del registro flag F; RLC A e RLCA non modificano gli stessi flag.

**4. RRC e RRCA:** analoghe a RLC e RLCA, con rotazione verso destra.

*Le tre operazioni per lo spostamento laterale dei bit sono:*

**1. SLA:** spostamento a sinistra "aritmetico".
Ciascun bit viene spostato a sinistra; il bit più significativo viene messo nel bit di riporto mentre quello meno significativo è posto a 0.
*Esempi:* SLA A
SLA (HL)

**2. SRL:** spostamento a destra "logico".
Ciascun bit viene spostato a destra, quello meno significativo viene messo nel bit di riporto e il più significativo viene posto a 0.
*Esempio:* SRL D

**3. SRA:** spostamento a destra "aritmetico".
Si tratta di un vero e proprio spostamento aritmetico in quanto il bit più significativo (quello che indica il segno) non viene modificato. Il byte viene spostato a sinistra di una posizione e il bit meno significativo viene messo nel bit di riporto.
*Esempi:* SRA A
SRA (HL)

Vi sono inoltre due istruzioni, RLD e RRD per operandi in decimale codificato in binario (B.C.D. = binary-coded decimal).

## Istruzioni sui bit

Ne esistono due, SET e RES, che pongono rispettivamente a 1 e a 0 il bit

di cui viene specificata la posizione.
*Esempio:* SET 4,A pone a 1 il quarto bit dell'accumulatore.
RES 6, (HL) pone a 0 il sesto bit del byte il cui indirizzo si trova nella coppia di registri HL.

L'istruzione BIT viene usata per controllare il valore di un bit in un registro o in una locazione di memoria. Il flag Z viene impostato con un valore inverso rispetto a quello del bit specificato, che non viene alterato.

## Istruzioni di salto e di sottoprogramma

Sei dei bit del registro flag F vengono effettivamente utilizzati; gli altri due sono vacanti:

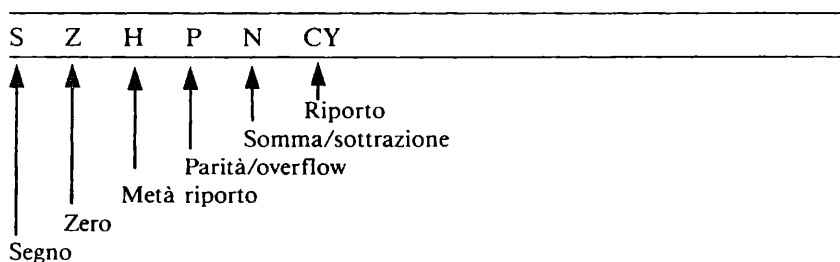



*Fig. 4.7 - Il registro flag F*

Molte delle operazioni svolte dallo Z-80 modificano, ponendoli di volta in volta a 0 o a 1, alcuni di questi flag in dipendenza del risultato dell'operazione stessa.
Il flag del segno S è una copia del bit piu significativo del risultato. Viene posto a 1 quando si presenta un valore negativo (in complemento a due).
Il flag Z viene posto a 1 solo quando il risultato è uguale a zero. Se nell'appendice D viene specificato che una delle operazioni modifica il flag Z, ne consegue che un risultato diverso da zero ne provoca l'azzeramento.
I flag di "metà riporto" e di "somma/sottrazione" vengono usati nelle operazioni tra valori in B.C.D. e sono di scarso interesse per i programmatori non specializzati.
Il flag di "parità/overflow" è insolito nel senso che viene posto a 1 o a 0 da certe istruzioni in accordo con un certo criterio, mentre altre istruzioni usano un criterio differente.
Se il flag viene modificato secondo la parità del risultato, esso viene posto

a 0 se il numero di 1 contenuti nel risultato stesso è dispari, altrimenti viene posto a 1. Usato come flag di overflow, esso viene posto a 1 se un'operazione di addizione o di sottrazione provoca una modifica erronea del bit 7. Nella rappresentazione binaria con segno tutti i valori negativi hanno il bit più significativo posto a 1. Un risultato fuori dall'intervallo di validità provoca un'inversione non voluta di questo bit. Il flag di overflow consente di accorgersi di una tale situazione.
Come abbiamo accennato in precedenza, il registro flag viene usato nelle istruzioni di salto condizionato. Se un particolare flag è posto a 0 o viceversa a 1, allora il salto viene effettuato; contrariamente il programma continua dall'istruzione successiva.
Le due principali istruzioni di salto sono JR (Jump Relative) e JP (JumP). Entrambe possono essere condizionate. L'istruzione JR viene accompagnata da un byte di spiazzamento il cui valore indica quanti byte devono essere saltati a partire dall'indirizzo contenuto nel contatore di istruzioni del programma. Lo spiazzamento può essere al massimo di 127 byte in avanti e 128 indietro. In caso di spiazzamento negativo, viene usata la rappresentazione in complemento a due.

*Esempi:* JP 40000 forza un salto alla locazione 40000 caricando semplicemente questo indirizzo nel contatore delle istruzioni del programma. JR —120 modifica il suddetto contatore in modo che, rispetto all'indirizzo della istruzione che sarebbe stata eseguita immediatamente dopo, esso punti 120 byte più indietro.
I codici di condizione che possono essere usati con entrambi i tipi di istruzione sono:

1. **Z e NZ:** JP Z,40000 esegue il salto alla locazione solo se Z è posto a 1. JR NZ,20 provoca il salto relativo solo se il bit Z del registro F è posto a 0.

2. **C e NC:** il salto viene eseguito se, rispettivamente, il bit di riporto è posto a 1 o a 0. L'istruzione JR può usare solo i codici di condizione Z, NZ, C e NC.

3. **PO e PE:** il salto viene effettuato se, rispettivamente, il bit di parità/overflow è posto a 1 o a 0. Questo codice di condizione può essere usato solamente dall'istruzione JP.

4. **P e M:** anche questi codici condizionali possono essere usati solo da JP e fanno in modo che il salto venga eseguito se, rispettivamente il bit S di segno del registro F è posto a 0 o a 1.

*Esempi:* JP P,22000 effettua un salto alla locazione 22000 se il flag di segno è posto a 0. JP PO,12000 invece effettua il salto solo nel caso sia posto a 1 il bit di parità/overflow.
Un'ultima istruzione di salto è la DJNZ che decrementa il contenuto del

registro B ed effettua il salto se il valore in B è diverso da 0. Per esempio DJNZ 20.
Il flag Z può essere posto a 1 o a 0 usando l'istruzione BIT. Questa infatti controlla uno specifico bit in un registro o in una locazione e se questo bit vale 0, il flag Z viene posto a 1, e viceversa. Per esempio BIT 4,A porrà a 1 il flag Z se il quarto bit dell'accumulatore vale zero.
L'attivazione di sottoprogrammai viene fatta con le istruzioni CALL e RET. Entrambe queste operazioni possono essere condizionate da una qualsiasi delle condizioni valide per la JP.

*Esempi:* CALL 20000 provoca il trasferimento del controllo dell'esecuzione al sottoprogramma che inizia all'indirizzo 20000, fino a che viene incontrata un'istruzione RET. L'indirizzo di ritorno viene memorizzato sullo stack. Un'attivazione di sottoprogramma con condizione ha la seguente sintassi: CALL NZ,12000 mentre un ritorno condizionato può essere RET Z.
Una variante dell'istruzione CALL è RST x, dove x deve essere un multiplo di 8 compreso tra 0 e 56. Quest'istruzione effettua semplicemente una chiamata al sottoprogramma all'indirizzo x.
Per esempio RST 24 effettua una chiamata all'indirizzo 24.

## Trasferimento di blocchi e operazioni di ricerca

Ci sono quattro istruzioni di trasferimento e quattro per la ricerca. Le istruzioni di trasferimento consentono di spostare blocchi di dati senza influenzare il contenuto dell'accumulatore:

**1. LDIR:** l'indirizzo del primo byte del blocco di dati che deve essere trasferito viene posto in HL mentre il numero di byte che lo compongono è posto in BC. Infine, prima che sia usata l'istruzione in oggetto, l'indirizzo di destinazione viene messo in DE. Ad ogni trasferimento di un byte BC viene decrementato e HL e DE incrementati. Il trasferimento termina quando BC diventa uguale a 0.
**2. LDDR:** essenzialmente è un'istruzione analoga alla LDIR, solo che il contenuto di HL e DE viene decrementato dopo il trasferimento di ogni byte.
**3. LDI:** analoga a LDIR, solo che viene trasferito un solo byte.
**4. LDD:** analoga a LDDR, solo che viene trasferito un solo byte.

*Esempio:* per copiare un blocco di 20 byte a partire dalla locazione 12000 nella locazione 14000.

```
LD HL,12000
LD BC,20
LD DE,14000
LDIR
```

Si potrebbe ottenere lo stesso risultato usando l'istruzione LDDR e caricando in HL 12019 ed in DE 14019.
Le istruzioni di ricerca controllano un blocco di dati fino a quando viene trovato un valore uguale a quello contenuto nell'accumulatore:

1. **CPIR:** la coppia BC viene caricata con la lunghezza del blocco su cui effettuare la ricerca e la coppia HL con l'indirizzo del primo byte del blocco stesso. Infine il termine di confronto, cioè il dato che viene verificato, viene posto nell'accumulatore. La ricerca termina quando BC vale zero oppure quando è stato trovato un byte il cui valore corrisponde a quello contenuto nell'accumulatore. Il flag Zero indica che:
Se posto a 1: è stata trovata corrispondenza tra l'accumulatore e un dato.
Se posto a 0: BC è stato decrementato fino a 0.
2. **CPDR:** istruzione analoga a CPIR con l'eccezione che la ricerca procede in memoria all'indietro, a partire dall'indirizzo contenuto in HL, invece che in avanti.
3. **CPI:** del tutto simile a CPIR, solo che viene controllato un solo byte.
4. **CPD:** analoga a CPDR, solo che viene controllato un solo byte.

Si noti che in tutte le istruzioni sia di trasferimento sia di spostamento il flag P/V indica se il conteggio ha raggiunto lo zero:
Se BC = 0 il P/V viene posto a zero.
Se BC è diverso da 0, allora P/V viene posto a 1.

*Esempio:* seguono le istruzioni necessarie per effettuare la ricerca di un byte con valore 4 nel blocco di memoria che parte dalla locazione 42000 e si estende per 120 byte.

```
LD BC,120
LD HL,42000
LD A,4
CPIR
```

Se al termine della ricerca il flag Z del registro F è posto a 1, allora la coppia di registri HL contiene l'indirizzo del byte cercato.

## Istruzioni di controllo dell'unità centrale e delle operazioni di ingresso/uscita

Il gruppo di istruzioni per il controllo dell'unità centrale può essere diviso in quattro parti:

1. Operazioni di scambio.

2. Le istruzioni NOP e HALT.
3. Le operazioni sullo stack.
4. Operazioni di ingresso/uscita e di interruzione.

**1. Operazioni di scambio**

Per effettuare degli scambi di valori esistono due istruzioni. La EXX scambia il contenuto di tre coppie di registri, la BC, DE e HL, con quelle equivalenti alternative, BC', DE' e HL'. La seconda istruzione, la EX, scambia semplicemente i valori contenuti in due registri a 16 bit. Quest'ultima ha quattro varianti:

**i.** EX AF,AF'.. scambia i registri A ed F con i registri alternativi A' ed F'.

**ii.** EX DE,HL .. scambia il valore della coppia DE con quello della coppia HL.

**iii.** EX (SP),IY e EX (SP), IX .. scambia il registro indice con i due byte che si trovano sulla cima dello stack.

**iv.** EX (SP), HL .. analogo al caso iii, tranne che ora è la coppia HL che prende il posto del registro indice.

**2. Le istruzioni NOP e HALT**

L'istruzione NOP (NO oPeration) non fa nulla per un periodo di 4 cicli macchina. Viene spesso usata per sostituire il codice ridondante o per riempire un'area richiesta per una futura espansione.
L'operazione HALT costringe il processore ad eseguire continuamente l'istruzione NOP fino a quando non si verifica un'interruzione o una rinizializzazione generale. Le NOP vengono effettuate per consentire allo Z-80 di aggiornare di continuo la memoria RAM. Infatti il registro che controlla quest'operazione viene automaticamente incrementato dopo che ogni istruzione è stata estratta dalla memoria per essere eseguita. Il valore che esso contiene rappresenta il byte meno significativo della sezione di memoria che deve essere aggiornata.

**3. Operazioni sullo stack**

Tutte le operazioni che concernono lo stack agiscono su dati a 16 bit. Il contenuto di ciascuna coppia di registri può essere messo sulla cima dello stack (PUSH) o viceversa caricato (POP) a partire dai due byte che si trovano in cima allo stack. La prima operazione non altera ovviamente il contenuto dei registri.
Dopo che l'operazione è stata completata, il registro puntatore allo stack contiene l'indirizzo del byte che si trova in cima allo stack. Quindi se il puntatore allo stack contiene X, il byte più significativo dell'ultima coppia di

registri messa sullo stack si trova all'indirizzo X — 1; quello meno significativo all'indirizzo X.
*Esempi:*

PUSH BC, POP BC, PUSH IY, POP IY

Il registro puntatore allo stack può essere accessibile direttamente usando le istruzioni LD, INC e DEC.
*Esempio:* INC SP:INC SP incrementerà il suddetto registro di due locazioni. Poiché lo stack si estende in memoria all'indietro, si sono persi i 16 bit che si trovavano sulla cima dello stack.

### 4. Operazioni di ingresso/uscita e di interruzione

Un'interruzione si verifica quando una componente esterna interrompe la normale attività esecutiva del processore. L'interruzione può essere provocata tramite una delle quattro linee di controllo dello Z-80:

1. BUSRQ
2. RESET
3. NMI
4. INT

Le linee 1, 2 e 3 sono raramente, anzi quasi mai, usate dal programmatore. La linea di BUSRQ, presentata in precedenza in questo capitolo, consente ad un altro processore di accedere al bus dati ed indirizzi del sistema. RESET viene usata per inizializzare il sistema al momento dell'accensione. I registri I ed R sono posti a zero prima che inizi l'esecuzione del programma a partire dalla locazione 0.
L'interruzione non mascherabile (NMI, dall'inglese Non Maskable Interrupt) costringe il processore ad eseguire una speciale routine alla locazione &H66. Nella configurazione MSX questa locazione è allocata dal sistema operativo per operazioni su disco e di conseguenza il NMI viene raramente implementato.
La linea INT può iniziare una qualsiasi di tre sequenze di operazioni. La scelta viene effettuata dall'utente con le istruzioni: IM0, IM1 o IM2.
In ogni caso il contenuto del registro contatore delle istruzioni del programma viene salvato sullo stack e viene fatto un salto alla locazione che viene indicata. Quando viene incontrata un'istruzione di RETI (RETurn from In-

terrupt), nel registro contatore delle istruzioni viene caricato l'indirizzo della locazione di ritorno e viene ripresa l'esecuzione dalla procedura che era stata interrotta.
La condizione di INT viene controllata dopo l'esecuzione di ogni istruzione. Tale condizione non è basata sul cambiamento di stato di una linea ma sul suo livello. Ne consegue che, a meno di voler attivare un'altra condizione di interruzione, il dispositivo da cui viene generata l'interruzione non deve continuare a tenere basso il livello della linea.
Il processore ha due circuiti bistabili (flip-flop) interni che caratterizzano il processo d'interruzione: IFF1 e IFF2. Se IFF1 è posto a 0, ogni richiesta di INT viene ignorata dal processore. Se viceversa è posto a 1, sono abilitate le interruzioni mascherabili.
Il programmatore, per mezzo delle istruzioni EI (Enable Interrupt = abilita le interruzioni) e DI (Disable Interrupt = disabilita le interruzioni), può porre a 1 o a zero sia IFF1 sia IFF2. IFF2 viene usato per memorizzare IFF1 durante un'interruzione non mascherabile, quando IFF1 è posto a uno per disabilitare ogni richiesta da INT.
L'indirizzo della procedura che deve essere seguita per "servire" l'interruzione viene indicato in modo differente per ciascuno dei modi della INT:

*Modo 0: Abilitato dall'istruzione IM0*

1. Sia IFF1 sia IFF2 sono poste logicamente a 0. Questo evita ogni problema di rientranza.

2. Lo Z-80 segnala la ricezione dell'interruzione al prossimo ciclo macchina tenendo basse due linee: IORQ e M1.

3. La componente che ha generato l'interruzione pone un codice operativo di un byte sul bus dati. Questo codice può essere sia una RST che una CALL..

4. Se viene indicato un codice operativo di RST, il contenuto del registro contatore di istruzioni del programma viene posto sullo stack e si esegue un salto alla locazione specificata nella pagina zero.

5. Dopo un codice operativo di chiamata, la componente da cui "sorge" l'interruzione deve porre altri due byte sul bus dati. Essi indicano l'indirizzo della routine che deve essere eseguita: l'esecuzione effettiva inizia dopo il salvataggio sullo stack del PC (program counter, vale a dire il registro contatore delle istruzioni del programma).

6. Quando viene incontrata un'istruzione di RETI, i due byte che si trovano sulla cima dello stack vengono caricati nuovamente nel PC. La RETI non pone a 1 il IFF1 e quindi le interruzioni devono essere riabilitate da una EI.

Si noti che deve essere usata anche un'istruzione di RET per assicurare una corretta ripresa della procedura la cui esecuzione era stata interrotta. Ciò nonostante certe componenti sono in grado di accorgersi quando lo Z-80 ha caricato per l'esecuzione un'istruzione di RETI. Questa capacità può essere sfruttata per rimuovere la richiesta iniziale di interruzione.

*Modo 1: Abilitato dall'istruzione IM 1*

La sequenza di inizializzazione che viene compiuta all'accensione da un elaboratore MSX seleziona questa modalità. La sequenza eseguita dopo una richiesta accettata di interruzione è la seguente:

1. IFF1 ed IFF2 vengono posti a zero.
2. Il contenuto del PC viene messo sullo stack.
3. L'esecuzione prosegue dalla locazione &H38.
4. Quando viene incontrata un'istruzione di RETI, i due byte di cima dello stack vengono rimessi nel registro contatore delle istruzioni, e viene quindi ripresa l'esecuzione del programma che era stato interrotto.

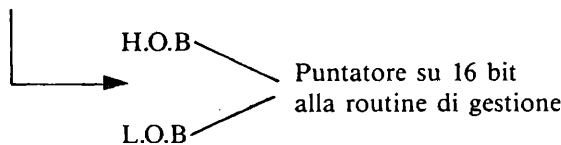
Si noti che per riabilitare le interruzioni mascherabili, è necessario porre a 1 IFF1 per mezzo dell'istruzione EI.

*Modo 2: Abilitato dall'istruzione IM 2*

La modalità 2 è la più flessibile delle tre consentite dalla INT. Permette infatti alla componente che genera l'interruzione l'accesso a 128 diverse procedure di gestione dell'interruzione stessa. Dopo che l'interruzione è stata ricevuta dal processore, la componente pone un solo byte sul bus dati. Questo viene unito al byte che si trova nel registro I in modo da formare un indirizzo su 16 bit:

Byte registro I / byte della componente
b b b b b b b b b b b b b b b b

Il puntatore su due byte indica quindi l'indirizzo della routine che deve essere eseguita.:

Indirizzo su 16 bit:

La sequenza eseguita è:

1. IFF1 viene posto a zero.
2. Lo Z-80 comunica la ricezione dell'interruzione tramite le linee IORQ e M1.
3. La componente che genera l'interruzione mette un byte sul bus dati.
4. L'indirizzo corrente contenuto nel registro PC viene posto sullo stack.
5. Viene eseguita la procedura di gestione che si trova all'indirizzo calcolato come visto sopra.
6. Quando viene incontrata una RETI, viene ripristinato il PC con i due byte in cima allo stack, e viene ripresa l'esecuzione del programma interrotto.

## Istruzioni di ingresso/uscita

In questo gruppo sono compresi due sottoinsiemi di istruzioni per effettuare operazioni di ingresso o di uscita da e su porte di 8 bit.

### 1. Ingresso/uscita di un solo byte

Le due istruzioni sono IN e OUT: ciascuna ha due formati possibili, il primo dei quali può usare solamente il registro A, mentre il secondo può far uso di uno dei seguenti registri: A, B, C, D, E, H o L.
Se si usa il formato particolare dell'accumulatore, la porta deve essere indicata tra parentesi: IN A,(porta). Nel secondo formato invece viene usato il registro C per selezionare la porta: OUT (C), registro.
Abbiamo già discusso in breve i meccanismi che regolano le operazioni di ingresso/uscita nella prima parte del capitolo. Lo Z-80 scrive il numero della porta sugli otto bit meno significativi del bus indirizzi prima di porre il dato sul bus dati.
Si noti però che sia nella istruzione IN A,(porta) che in OUT (porta),A, il processore scrive il contenuto dell'accumulatore sulle linee dalla A15 alla A8. In modo analogo le istruzioni IN registro,(C) e OUT (C), registro implicano la scrittura del registro B sulle linee di indirizzo superiore. Ciò consente di utilizzare più di 256 porte, a patto di disporre del necessario hardware di decodifica.

### 2. Istruzioni di ingresso/uscita per blocchi di dati

I blocchi di dati possono essere formati da un minimo di un byte fino a un massimo di 256 byte. In questo gruppo di istruzioni ci sono quattro coppie

di codici operativi, in modo analogo a quanto accade per i gruppi di istruzioni di trasferimento e di ricerca:

**i.** INIR e OTIR: i dati di inizializzazione richiesti sono:

a. il numero della porta nel registro C.
b. l'indirizzo del primo dato in HL.
c. la dimensione del blocco nel registro B.

La coppia di registri HL viene incrementata dopo ogni trasferimento. Se il registro B è stato decrementato fino a zero, viene posto a 1 il flag Z.

**ii.** INDR e OTDR: analoghe alle due precedenti, con la sola differenza che HL viene decrementato dopo ogni trasferimento.

**iii.** INI e OUTI: analoghe alle istruzioni INIR e OTIR, solo che provocano l'ingresso o l'uscita di un solo byte. HL viene incrementato mentre B viene decrementato.

**iv.** IND e OUTD: analoghe a INI ed a OUTI, ad eccezione del fatto che la coppia HL viene decrementata ad ogni trasferimento.

## Modalità d'indirizzamento

Come abbiamo avuto modo di vedere, ciascun codice operativo consente di specificare in diversi modi l'indirizzo del dato su cui si deve operare. Benché lo Z-80 abbia dieci diverse modalità d'indirizzamento, non è necessario che ogni tipo d'istruzione abbia una variante per ciascuna delle modalità. Il lettore troverà autoesplicativa la maggior parte dei formati - molti dei quali sono già stati presentati:

**1. Implicito**: Non viene specificato alcun indirizzo, in quanto è implicito nell'istruzione.

Esempio: NEG.

**2. Immediato**: Viene specificato, invece dell'indirizzo, il dato corrente, su 8 bit.

Esempio: LD A,2

**3. Immediato esteso**: Viene specificato, invece dell'indirizzo, il dato corrente, su 16 bit.

Esempio: LD HL,20000

**4. Relativo**: Questa modalità viene usata solamente dall'istruzione di salto relativo e da DJNZ. Il byte che segue il codice operativo ha un valore con segno compreso tra —128 e 127. Questo indica lo spiazzamento che deve essere sommato all'indirizzo contenuto nel contatore di istruzioni del programma per ottenere l'indirizzo della prossima istruzione che deve essere eseguita.

Esempio: JR 20

**5. Registro**: Il dato su cui operare è contenuto nel registro che viene specificato.

Esempio: LD A,B

**6. Registro indiretto**: Un registro indice o una coppia di registri contengono l'indirizzo del dato su cui operare. Il registro o la coppia di registri vengono indicati tra parentesi.

Esempio: LD A,(DE)

**7. Esteso**: L'indirizzo del dato viene indicato tra parentesi.

Esempio: LD A,(4000)

**8. Indiciato**: Si tratta di una forma essenzialmente analoga a quella "registro indiretto, con l'aggiunta di uno spiazzamento che deve essere aggiunto al valore contenuto dal registro indice:

Esempio: LD A,(IY+5)

**9. Pagina zero modificata**: Solo la RST usa questa modalità. Viene effettuata una chiamata alla locazione specificata. Quest'ultima deve essere un multiplo di 8 compreso tra 0 e 56.

Esempio: RST 8

**10. Bit**: La modalità bit è insolita in quanto non viene specificato un byte, bensì un bit. Viene usata da tre tipi di istruzioni: BIT, SET e RES. I bit sono numerati come segue:

```
X  X  X  X  X  X  X  X
7  6  5  4  3  2  1  0
```

Esempio: SET 4,A

Si è così arrivati al momento in cui i lettori possono provare a scrivere loro

stessi delle routine in linguaggio macchina. I seguenti consigli potranno dimostrarsi utili a tale proposito:

1. Salvate sempre un programma scritto in codice macchina prima di provarne il funzionamento. Potrebbe infatti contenere un errore: in tal caso la macchina si bloccherebbe; sareste così costretti a spegnerla, perdendo il codice, per ripristinare la situazione.

2. È improbabile che una routine funzioni correttamente al primo colpo. Se siete fortunati due o tre sue parti funzioneranno nel modo voluto: ci vuole pazienza.

3. Se volete unire un codice macchina a un programma BASIC, è più semplice usare un ciclo in BASIC per caricare il codice macchina nella posizione voluta, piuttosto che caricare separatamente i programmi per poi salvarli. Se i due programmi devono passarsi dei valori, è preferibile usare delle istruzioni PEEK e POKE piuttosto che la funzione USR.

4. In generale le routine scritte in linguaggio macchina richiedono una fase di messa a punto più lunga: vale quindi la pena di dividere i programmi in piccole sezioni, ciascuna delle quali viene attivata da un programma principale.

Capitolo 5
# LA CONFIGURAZIONE MSX

La specifica di base per i sistemi MSX comprende un'unità centrale di elaborazione Z-80 o equivalente, funzionante a 3.5 MHz, affiancata da un processore video TMS-9929A della Texas Instruments (TMS-9918A in America e Giappone) e da un integrato per la generazione di suoni AY-3-8910 della General Instruments.
Oltre ai suddetti componenti il sistema prevede anche un'interfaccia programmabile 8255 della Intel, che gestisce la scansione della tastiera e il sistema di paginazione della memoria. La specifica viene poi completata dai 32K del BASIC MSX e dalla ROM del sistema operativo e da un minimo di 8Kbyte di RAM utente (anche se sul mercato europeo sembra assai improponibile la vendita di una macchina con meno di 32K). Inoltre il processore video viene fornito di 16K di RAM dedicata al video che viene usata per memorizzare lo schermo, i colori e la definizione degli sprite. L'uso di questa 'VRAM' verrà discussa più a fondo nel Capitolo 6.

## Gestione della memoria nell'MSX

Il concetto di "partizione" (è un concetto logico; fisicamente viene realizzato con più banchi di memoria che trovano posto in diversi alloggiamenti, o slot) è fondamentale per la comprensione della gestione della memoria nei sistemi MSX: tale concetto verrà ora presentato in breve.
Poiché lo Z-80 è un processore a soli 8 bit, il suo massimo spazio d'indirizzamento è di 64K byte; però il sistema MSX è stato progettato per essere in grado di accedere fino a 1024K byte: di conseguenza è stato adottato il sistema delle "partizioni".

Una "partizione" è uno spazio d'indirizzamento di 64Kbyte: il sistema MSX può indirizzarne fino a quattro. Ciascuna di queste "partizioni primarie" può a sua volta essere divisa in quattro "partizioni" secondarie dando così luogo ad uno spazio indirizzabile complessivo di 4 * 4 * 64 = 1024 Kbyte. Ciascuna partizione (primaria o secondaria) è a sua volta divisa in quattro pagine di 16 Kbyte ciascuna: queste ultime sono le entità di base che il sistema MSX usa per creare il suo spazio d'indirizzamento. Qualsiasi sia la partizione di cui si sta trattando, le quattro pagine iniziano rispettivamente agli indirizzi 0000H, 4000H, 8000H e 0C000H. Le pagine possono essere indirizzate solo ed unicamente al loro indirizzo reale (cioè pagina 0 a 0000H, pagina 1 a 4000H, ecc.). In tal modo il sistema MSX può accedere a qualsiasi pagina di una qualsiasi partizione, ma deve comunque fare sempre riferimento per la pagina 0 all'indirizzo 0000H, per la pagina 3 a 0C000H e così via.
La selezione della partizione primaria viene effettuata scrivendo nella porta A dell'8255 PPI nel seguente formato:

Poiché l'allocazione su ciascuna partizione viene descritta con 3 bit, ne consegue che ogni allocazione può essere indicata numericamente con un valore compreso tra 0 e 3 (00B e 11B), consentendo così la selezione di uno qualsiasi delle quattro partizioni principali per ognuna delle quattro pagine. La selezione di una partizione secondaria si ottiene scrivendo, nello stesso formato, alla locazione 0FFFFH dell'alloggiamento principale. Quando si effettua una lettura da questo indirizzo il valore ritornato è il complemento di quello reale.
Al momento dell'accensione (partenza "a freddo") il sistema MSX (che è residente nella partizione 0 che è quindi il primo ad essere accessibile dalla CPU) inizia una ricerca dell'area contigua di RAM più ampia a partire dall'indirizzo 0FFFFH verso il basso e alloca quest'area per le pagine 3 e 2. In condizioni normali, quindi, non è necessario che il programmatore effettui esplicitamente una scelta della partizione, tranne il caso di un sistema con una RAM di 64 Kbyte o più, laddove c'è sovrapposizione tra RAM e ROM, o nel caso di un software basato su cartuccia che è stato scritto per essere eseguito in partizioni diverse da 0.

## Metodi di accesso al generatore di suoni, al VDP e alla PPI

Lo Z-80, oltre a poter indirizzare 64 Kbyte di memòria, può anche accedere agli indirizzi di 256 porte di ingresso/uscita. Questi ultimi vengono utilizzati per accedere al VDP e alle altre componenti del sistema (comprese l'interfaccia RS-232 e le altre espansioni previste). Però, per permettere eventuali modifiche hardware, le chiamate al BIOS (Basic I/O System = sistema di base per l'ingresso/uscita) vengono fornite nella ROM del sistema MSX. Questo consente di accedere alle componenti del sistema in modo tale che se in una unità MSX un integrato occupa una posizione differente, ciò viene compensato dal firmware. La differenza è quindi invisibile da parte del programmatore e viene così assicurata una completa compatibilità software. L'unica eccezione a questa regola è il VDP (il processore video), che sarà preso in seguito approfonditamente in esame.

### Introduzione generale al processore video

Il processore video TMS-9929A della Texas Instruments è a sua volta un microprocessore. Esso è responsabile dell'intera gestione del video ed accede a 16K di RAM video dedicata. In quest'ultima sono contenute le definizioni dei caratteri, i dati di schermo, i dati di colore, la definizione degli sprite e i loro attributi. In tal modo la RAM di sistema è libera di essere usata per la programmazione (quel che si perde è una parte della velocità per accedere alla RAM video). In circostanze normali, però, questa riduzione non è critica. In situazioni critiche, viceversa, esiste la possibilità di intervenire con vari "trucchi", che verranno discussi nel Capitolo 6.
Il VDP ha quattro modalità di visualizzazione: Grafica I, Grafica II, Multicolore e Testo. La modalità testo fornisce 24 righe di 40 colonne con due colori ed è stata progettata per massimizzare le capacità di visualizzazione sui video dei televisori domestici. La modalità multicolore fornisce un video di 64 * 48 punti con 15 colori più il trasparente. La modalità Grafica I fornice un video da 256 * 192 pixel per la generazione di grafici in 15 colori più il trasparente. La modalità Grafica II è un miglioramento della Grafica I in quanto consente di ottenere colori e figure più complesse.
Il display video consiste di 34 piani che devono essere considerati come sovrapposti in profondità. Quello più interno è il piano dello sfondo, quello successivo è il piano su cui agiscono le modalità Grafica I e II, mentre i restanti sono i 32 piani riservati agli sprite.
Oltre a gestire il video, il VDP si occupa del "refresh" della memoria RAM utente ed effettua una richiesta d'interruzione ogni 20 ms. Per avere ulteriori dettagli sul VDP e sulla sua programmazione fate riferimento al Capitolo 6.

### Il generatore di suoni General Instruments AY-3-8910

Il GI AY-3-8910 è un integrato per la generazione di suoni, a tre voci con un estensione di otto ottave; comprende anche un semplice controllo hardware della curva di volume ed è in grado di unire un rumore "bianco" a ciascuna delle tre voci. L'integrato comprende inoltre due porte di ingresso/uscita che vengono sfruttate dal sistema per i joystick e le tavolette tattili. Per una descrizione delle modalità di programmazione di questa componente si faccia riferimento al Capitolo 7.

### La PPI Intel 8255

Il PPI Intel 8255 è un integrato per l'I/O potente e d'utilizzo generale che, nel sistema MSX, viene usato per diversi scopi. È sua responsabilità la gestione della tastiera (comprese quelle di estensione), varie funzioni di ingresso/uscita per le cassette, il sistema di gestione della memoria e fornisce una porta sonora di 1 bit.
Per una descrizione più completa dell'integrato e dei suoi vari usi si faccia riferimento al Capitolo 8.

## La gestione delle interruzioni e gli "agganci" della RAM

Molte procedure che si trovano nella ROM dell'MSX vengono ridirette per mezzo di "agganci" nella RAM. Un "aggancio" consiste in 5 byte di RAM inizializzati in modo da contenere un'istruzione RET dello Z-80 (0C9H). La ridirezione viene effettuata dalla procedura in questione, che si trova in ROM, per mezzo di una chiamata (CALL dello Z-80) riferita all'aggancio. Diventa quindi semplice ridirigere la chiamata inserendo un'istruzione Z-80 di "JP nnnn" nei primi tre byte dell'"aggancio".
Il sistema MSX funziona secondo la modalità 1 d'interruzione e rende disponibili due "agganci" per la gestione delle IRQ (richieste d'interruzione). Il primo di questi si trova alla locazione 0FD9AH e consente di gestire delle richieste d'interruzione che vengono originate da dispositivi diversi dal temporizzatore del VDP. Nella versione di base dell'MSX questo "aggancio" è di scarso utilizzo, ma è stato previsto per future espansioni. Il secondo è alla locazione 0FD9FH e viene usato per la gestione delle interruzioni generate dal segnale di sincronismo video del VDP: si tratta dell'unica interruzione disponibile nel sistema di base. L'interruzione del VDP infatti gestisce anche la scansione della tastiera e varie altre funzioni del sistema operativo. È essenziale, a questo proposito, che una qualsiasi procedura utente restituisca il controllo alla routine di sistema operativo che l'aveva atti-

vata: in tal modo viene mantenuta una corretta gestione dell'interruzione in questione.
Il sistema operativo salva tutti i registri (compresi quelli alternativi) sullo stack prima di effettuare la chiamata alla locazione 0FD9FH: quindi tutti i registri possono essere utilizzati senza preoccupazione alcuna. Però il sistema operativo porta anche nell'accumulatore il contenuto del registro di stato del VDP prima di chiamare l'"aggancio" ed è quindi essenziale che nessuna routine modifichi la coppia di registri AF. Al ritorno dall'"aggancio" il sistema operativo memorizza il valore dell'accumulatore all'indirizzo 0F3E7H: quando si debba testare uno dei bit del registro di stato del VDP all'interno di una procedura utente non attivata da interruzione, è preferibile leggere questa locazione piuttosto che il registro di stato del VDP direttamente.
Oltre agli agganci di cui si è detto, il sistema ne fornisce un altro all'indirizzo 0FDD6H per la gestione delle interruzioni non mascherabili (NMI). Ciò è però apparentemente di scarsa utilità, in quanto da un lato il sistema di base non prevede nessuna interruzione non mascherabile e dall'altro perché nel sistema di gestione del disco il vettore d'entrata associato alle NMI, all'indirizzo 66H, è occupato dal blocco di controllo del file per il DOS. Nelle applicazioni utente l'uso di NMI può rivelarsi necessario e quindi l'aggancio in questione viene reso disponibile per aumentare la flessibilità generale del sistema.
Il seguente programma realizza un temporizzatore in tempo reale attivabile tramite un'interruzione: esso costituisce un esempio di gestione delle richieste d'interruzione (IRQ) da parte dell'utente.

**Esempio di temporizzatore in tempo reale**

```
WRTVDP  EQU 47H     ;CHIAMA LA TABELLA DEI BIOS
RDVRM   EQU 4AH
WRTVRM  EQU 4DH
FILVRM  EQU 56H
LDIRVM  EQU 5CH
LDIRMV  EQU 59H
CHGET   EQU 9FH
CHPUT   EQU 0A2H
GTSTCK  EQU 0D5H
GTTRIG  EQU 0D8H
RDVDP   EQU 13EH
SNSMAT  EQU 141H    ;FINE DELLA TABELLA DEI BIOS
```

```
        VDPRGS EQU 0F3DFH ;L'AREA DI SALVATAGGIO MSX
                          ;PER I REGISTRI VDP
        BASNOS EQU 0F3B3H ;INIZIO AREA PER LE TABELLE
                          ;VDP
        INTHOK  EQU 0FD9FH
        ORG 0E000H
START:  LD HL,MESS        ;POSIZIONA HL AL PUNTO
                          ;DI INPUT
MPLP:   LD A,(HL)
        CP '$'
        JR Z,GETIN
        CALL CHPUT        ;PRONTO PER LA STAMPA
        INC HL
        JR MPLP
GETIN:  LD HL,HHMMSS
        LD B,3
GTINOL: CALL GETNUM       ;PRENDE 2 SEMIBYTE NUMERICI
        LD (HL),A         ;MEMORIZZA IN HHMMSS
GTINIL: CALL CHGET        ;PRENDE I CARATTERI
        CP ':'            ;È IL SEPARATORE ?
        JR NZ,GTINIL      ;SE NO PROVA ANCORA
        INC HL            ;PUNTA AL DATO SUCCESSIVO
                          ;IN HHMMSS
        DJNZ GTINOL       ;LO FA TRE VOLTE
        LD A,LOW CLOCK    ;SET UP...
        LD (INTHOK+1),A   ;INDIRIZZO DI JP
        LD A,HIGH CLOCK
        LD (INTHOK+2),A
        LD A,0C3H         ;='JP'DI Z80
        LD(INTHOK),A
        RET               ;AL BASIC, AVENDO RIDIRETTO
                          ;L'AGGANCIO
GETNUM: CALL CHGET        ;PRENDE UN CARATTERE
        CP'0'
        JR C,GETNUM
        CP ':'            ;È NUMERICO?
        CALL CHPUT        ;LO STAMPA
        SUB '0'           ;LO CONVERTE IN NUMERICO
        SLA A
        SLA A
        SLA A             ;MOLTIPLICA PER 16
        SLA A
```

```
         LD B,A              ;LO METTE IN B
GTNUM1:  CALL CHGET          ;PRENDE UN CARATTERE
         CP'0'
         JR C,GETNUM1
         CP ':'              ;È NUMERICO?
         JR NC,GETNUM1       ;SE NON ANCORA PROVATO
         CALL CHPUT          ;LO STAMPA
         SUB '0'             ;LO CONVERTE IN NUMERICO
         ADD A,B             ;LO SOMMA A B
         RET                 ;RITORNA COL NUMERO IN A
MESS:    DEFB 'ENTER TIME HH:MM:SS:$'
HHMMSS:  DEFB 0,0,0
CTR:     DEFB 0
CLOCK:   PUSH AF             ;CONSERVA LO STATO DEL VDP
         LD A,(CTR)
         DEC A
         LD (CTR), A         ;ATTIVA L'OROLOGIO OGNI 50 MS
         JP P,OUT            ;INTERRUZIONE
         LD A,49
         LD (CTR),A          ;RIPOSIZIONA IL CONTATORE
         LD IX,HHMMSS
         LD A,(IX+2)
         INC A               ;INCREMENTA I SECONDI
         DAA
         LD (IX+2),A
         CP 60H              ;SE NECESSARIO...
         JR NZ,CLKPNT
         XOR A               ;RIPOSIZIONA I SECONDI
         LD (IX+2),A
         LD A,(IX+1)
         INC A               ;ED INCREMENTA I MINUTI
         DAA
         LD (IX+1),A
         CP 60H              ;SE NECESSARIO...
         JR NZ,CLKPNT
         XOR A               ;RIPOSIZIONA I MINUTI
         LD (IX+1),A
         LD A,(IX+0)
         INC A               ;ED INCREMENTA LE ORE
         DAA
         LD (IX+0),A
         CP 24H
```

```
         JR NZ,CLKPNT
         XOR A
         LD (IX+0),A
CLKPNT:  LD HL,VDPRGS          ;ROUTINE PER STAMPARE
         LD A,(HL)             ;OROLOGIO AGGIORNATO
         LD DE,10              ;LEGGE COPIE NELLA RAM
                               ;DI SISTEMA
         AND 2                 ;DEI REGISTRI A SOLA
                               ;SCRITTURA
         JR NZ,GNAMTB          ;DEL VDP.
         INC HL                ;PER TROVARE LA MODALITÀ
                               ;DELLO SCHERMO
         LD A,(HL)             ;E CALCOLARE L'OFFSET...
         AND 8                 ;PARTENDO DALLA BASE (N)...
         JP NZ,OUT             ;TABELLA VARIABILE.
         LD A,(HL)             ;PER TROVARE L'INDIRIZZO
                               ;DI BASE
         AND 16                ;DELLA TABELLA DEI NOMI.
         LD DE,0
         JR NZ,NGAMTB
         LD DE,5
GNAMTB:  LD IX,BASNOS
         ADD IX,DE
         LD L,(IX+0)           ;RICAVA L'INDIRIZZO DELLA
                               ;TABELLA DEI NOMI
         LD H,(IX+1)
         LD DE,24              ;ED AGGIUNGE 24(OFFSET)
         ADD HL,DE
         LD IX,HHMMSS          ;DÀ L'INDIRIZZO DEL TEMPO
         LD A,(IX+0)           ;DÀ IL VALORE PRINCIPALE
                               ;(ORE)
         CALL NPNT             ;LO STAMPA
         LD A,':'              ;CARICA I DUE PUNTI
         CALL WRTVRM           ;LI STAMPA
         INC HL                ;INCREMENTA LA POSIZIONE
                               ;DI STAMPA
         LD A,(IX+1)           ;RIPETE PER I MINUTI
         CALL NPNT
         LD A,':'
         CALL WRTVRM
         INC HL
         LD A,(IX+2)           ;E PER I SECONDI
```

```
        CALL NPNT
OUT:    POP AF          ;RIPORTA VDP ALLO STATO
        RET             ;INIZIALE E RITORNA
                        ;ALLA ROUTINE OS
NPNT:   PUSH AF         ;SALVA IL VALORE NUMERICO
        SRL A           ;CONSIDERA IL SEMI-BYTE
                        ;DI TESTA
        SRL A
        SRL A
        SRL A
        ADD A,30H       ;CONVERTE IN CODICE ASCII
        CALL WRTVRM     ;LO STAMPA
        INC HL          ;INCREMENTA LA POSIZIONE
                        ;DI STAMPA
        POP AF          ;RIPRISTINA IL VALORE
                        ;NUMERICO
        AND 15          ;CONSIDERA L'ALTRO SEMIBYTE
        ADD A,30H       ;CONVERTE IN CODICE ASCII
        CALL WRTVRM     ;LO STAMPA
        INC HL          ;INCREMENTA LA POSIZIONE
                        ;DI STAMPA
        RET

        END
```

## Uso della RAM di sistema dell'MSX

Nel sistema MSX le locazioni da 0F380H a 0FFFFH sono usate dal BASIC e dal sistema operativo per svolgere varie funzioni di gestione generale. Alcune di queste locazioni particolarmente utili a chi programma nel linguaggio Assembly verranno descritte in seguito.
Le procedura di lettura e scrittura da una partizione all'altra e le routine di chiamata vengono fornite all'inizio della RAM di sistema.
All'indirizzo 0F380H inizia una routine che consente di leggere da una qualsiasi delle partizioni primarie. Quest'ultima ha in ingresso un valore per la selezione della partizione (nel formato richiesto dal 8255) che deve essere posto nell'accumulatore; nel registro D viene posto il vecchio valore dello stato della partizione e nella coppia HL viene posto l'indirizzo. La rou-

tine ritorna il valore letto nel registro E, lasciando invariati tutti i rimanenti registri.
Una routine analoga a quella appena descritta inizia all'indirizzo 0F385H: essa agisce con le medesime convenzioni sul contenuto dei registri, ma effettua una scrittura del valore contenuto in E.
Infine la routine che si trova in 0F38CH effettua chiamate da una partizione all'altra. Questa procedura si aspetta di trovare sullo stack il vecchio valore di stato della partizione e quello nuovo nell'accumulatore; inoltre un valore che debba essere passato deve essere nella coppia di registri alternativi AF, e ancora l'indirizzo a cui fare la chiamata in IX. Ogni valore ritornato dovrà essere messo nella coppia di registri alternativi AF.
Il sistema memorizza gli indirizzi di chiamata per la funzione USR del BASIC nelle locazioni dalla 0F39AH alla 0F3ADH ordinatamente dalla USR0 fino alla USR9, allocando due byte per ciascuna. Le locazioni dalla 0F3B3H alla 0F3D9H contengono i valori che vengono messi in corrispondenza alla pseudovariabile BASE(N) del BASIC, predisponendo due byte per ciascun valore.

Nella locazione 0F3DBH viene memorizzata l'opzione "key click" (attivando la quale si ottiene un piccolo segnale sonoro quando si premono i tasti): scrivendo 0 in questo indirizzo l'opzione viene disabilitata; viceversa per un valore diverso da zero.
La posizione, in coordinate xy, del cursore sullo schermo viene memorizzata agli indirizzi 0F3DCH (Y) e 0F3DDH (X).
Le otto locazioni dalla 0F3DFH in avanti vengono usate dal sistema operativo per memorizzare i valori attuali degli otto registri a sola scrittura del VDP. Il valore attuale del registro di stato del VDP viene memorizzato durante ogni intervallo di sincronismo verticale nella locazione 0F3E7H. Le locazioni dalla 0F3E9H alla 0F3EBH vengono usate per memorizzare i colori correnti per lo sfondo, il primo piano e il bordo dello schermo, così come vengono impostati tramite il comando COLOR del BASIC.

Il sistema memorizza l'indirizzo più alto che trova nella RAM nelle locazioni 0F672H e 0F673H. Questi indirizzi possono essere cambiati per "nascondere" alcune zone della RAM al BASIC e al sistema operativo, in modo da aumentare la sicurezza del codice messo in queste aree. I due byte che seguono questi indirizzi definiscono l'indirizzo più alto che può essere usato dallo stack.

Le 26 locazioni che partono dall'indirizzo 0F6CAH vengono usate per ricordare i tipi di default delle variabili con inziale dalla A alla Z. Questa tabella viene modificata con delle istruzioni DEFINT, DEFSTR, DEFSNG ecc. ed è letta quando viene incontrata una qualsiasi variabile non accompagnata da un suffisso che ne indichi esplicitamente il tipo. I tipi delle variabili sono:

*Intero:* valore in tabella 2.
*Stringa:* valore in tabella 3.
*Singola precisione:* valore in tabella 4.
*Doppia precisione:* valore in tabella 8.

Si noti che il default per tutte le entrate della tabella è sempre "doppia precisione".
Infine, dalla 0FD9AH alla 0FFCAH, troviamo il blocco degli "agganci" di ridirezione alla RAM ciascuno dei quali consiste in 5 byte inizializzati con l'istruzione Z-80 RET (0C9H). La maggior parte di questi "agganci" sono stati previsti per future espansioni del sistema e risultano di limitata utilità per il programmatore. Quelli invece di uso immediato verranno discussi quando li si incontrerà nell'esaminare il resto del sistema.

## Chiamate da BASIC di sottoprogrammi in codice macchina

I segreti per interfacciare correttamente procedure scritte in codice macchina con il BASIC sono essenzialmente due: prima di tutto "nascondere" la routine in codice macchina in modo che il BASIC non possa, per errore, sovrascrivere nella sua area, e, seconda cosa, passare alla routine stessa dei parametri e leggerne i risultati.
Il metodo più semplice per riservare spazio alle procedure in codice macchina è l'uso dell'istruzione BASIC CLEAR n1, n2. Il primo parametro indica lo spazio riservato per la memorizzazione delle stringhe, mentre il secondo determina l'indirizzo più alto che può essere acceduto dal BASIC. Quindi per riservare i 16K più alti della RAM al codice macchina e per predisporre uno spazio di 200 byte per le stringhe è necessario usare il comando CLEAR 200,&HBFFF. Si noti però che in circostanze normali l'area di lavoro del sistema rimane quella dalla locazione 0F380H in avanti.
Per chiamare routine in codice macchina dal BASIC si deve fare ricorso alla funzione USRn; la sintassi di quest'ultima è:

```
Variabile = USRn(variabile/costante) oppure
PRINT USRn(variabile/costante)
```

dove n è il numero della routine utente definito dalla DEFUSR e sia variabile sia variabile/costante possono essere di un tipo qualunque.
Quindi se vogliamo passare una stringa ad una routine che deve ritornare un valore intero, possiamo chiamarla così:

```
A% = USR1("abcde")
```

La chiamata alla USR effettua un passaggio dei valori secondo la seguente modalità:

*Intero:* la locazione 0F663H.contiene 2, e il valore è memorizzato in 0F7F8H e 0F7F9H (il primo è il byte meno significativo).

*Stringa:* la locazione 0F663H contiene 3, e 0F7F8H e 0F7F9H contengono l'indirizzo del descrittore della stringa. Un descrittore di una stringa è composto da tre byte: la sua lunghezza e l'indirizzo dove è effettivamente memorizzata la stringa stessa.

*Singola precisione:* la locazione 0F663H contiene 4, e il valore è memorizzato nella locazioni dalla 0F7F6H alla 0F7F9H.

*Doppia precisione:* la locazione 0F663H contiene 8, ed il valore è memorizzato nelle locazioni dalla 0F7F6H alla 0F7FDH.

I parametri possono essere ritornati al BASIC in una maniera altrettanto semplice. Il programma che segue illustra questo problema: viene acquisita una stringa numerica che rappresenta un valore in base tre, e viene ritornato come intero il decimale equivalente.

```
        VARTYP EQU 0F663H
        VARPTR EQU 0F7F8H
        ORG 09000H
START:  LD A,(VARTYP)
        CP 3                ;VAR È UNA STRINGA?
        RET NZ              ;SE NON LO È RETURN
        LD HL,(VARPTR)      ;PRENDE L'INDIRIZZO
                            ;DEL DESCRITTORE
        LD B,(HL)           ;LUNGHEZZA DELLA STRINGA
                            ;IN B
        INC HL
        LD E,(HL)           ;INDIRIZZO DELLA STRINGA
                            ;IN DE
        INC HL
        LD D,(HL)
        LD HL,0             ;VUOTA HL
ICLP:   PUSH DE
        PUSH HL             ;SALVA I REGISTRI
        ADD HL,HL
        POP DE
        ADD HL,DE           ;VALBIN=VALBIN*3
        POP DE              ;PRENDE IL PUNTATORE ALLA
                            ;STRINGA CORRENTE
```

```
EX DE,HL            ;IN HL E VALBIN IN DE
LD A,(HL)           ;PRENDE DALLA STRINGA
                    ;I CARATTERI SUCCESSIVI
SUB '0'             ;LI CONVERTE IN NUMERICO
INC HL              ;INCREMENTA LA STRINGA PTR
PUSH HL             ;LA SALVA
LD L,A
LD H,0              ;METTE IL NUMERO IN HL
ADD HL,DE           ;VALBIN=VALBIN+NUM
POP DE              ;PUNTATORE ALLA STRINGA
                    ;IN DE
DJNZ ICLP           ;SE VI SONO ANCORA
                    ;STRINGHE LO RIPETE
LD (VARPTR),HL      ;VALBIN IN VARPTR
LD A,2
LD (VARTYP),A       ;VARTYP=INTEGER
RET                 ;AL BASIC

END
```

## Capitolo 6

# IL PROCESSORE VIDEO

Il processore video (VDP, dall'inglese Video Display Processor) si interfaccia con la CPU attraverso un bus dati bidirezionale su 8 bit, tre linee di controllo e un'interruzione. Si possono effettuare quattro tipi di operazioni: scrittura di dati nella VRAM, lettura di dati dalla VRAM, scrittura dei registri di controllo del VDP e lettura del registro di stato del VDP stesso. Ciascuna di queste operazioni implica uno o più trasferimenti di dati sul bus d'interfaccia tra il VDP e la CPU, dove l'interpretazione di questi trasferimenti è legata allo stato delle tre linee di controllo. La CPU può comunicare con il VDP in modo asincrono rispetto alla scansione video della televisione poiché il VDP consente l'accesso alla VRAM anche durante il periodo in cui viene effettuata la scansione video.

## Le linee di controllo

Le tre linee di controllo (CSW, CSR e MODE) determinano l'interpretazione dei trasferimenti dati da parte del VDP. La linea CSW è attiva (bassa) quando è in corso un trasferimento dalla CPU al VDP. La linea CSR è attiva (bassa) quando deve essere fatta una lettura da parte della CPU. Le due suddette linee non sono mai attive simultaneamente.
La linea MODE indica la sorgente o la destinazione di un'operazione di lettura o di scrittura, e, nella attuale configurazione del MSX, viene collegata alla linea 0 del bus indirizzi della CPU. La figura 6.1 illustra globalmente le operazioni d'interfaccia tra VDP e CPU, che vengono descritte in seguito.

### La CPU scrive nei registri del VDP

Il VDP ha otto registri di controllo a sola scrittura e un registro di stato a sola lettura.
Ciascuno degli otto registri a sola scrittura può essere caricato da parte della CPU con due soli trasferimenti di dati su 8 bit (lo stato della linea di controllo per ottenere questa e tutte le altre operazioni si trova nella tabella 6.1). Il primo byte trasferito è il dato che deve essere scritto, mentre con il secondo si specifica il registro destinazione (da 0 a 7, decimale: il bit più significativo a 1 indica una scrittura di registro; se posto a 0 indica invece un'operazione di configurazione dell'indirizzo della VRAM).
Per riscrivere dati nei registri interni dopo che è già stato trasferito un byte dato, è necessario leggere il registro di stato del VDP per poter rinizializzare la logica dell'interfaccia. Questa situazione è abbastanza frequente in ambienti operativi, come quello del MSX, guidati da interruzioni. In generale, qualora sia dubbio lo stato dei parametri di lettura/scrittura del VDP, è opportuno svolgere la suddetta procedura.

### La CPU scrive nella VRAM

Il trasferimento dati dalla CPU alla VRAM viene effettuato utilizzando un registro d'indirizzamento su 14 bit che si autoincrementa. I primi due byte trasferiti in scrittura alla VRAM vengono usati per impostare correttamente il suddetto registro: le scritture successive richiedono quindi il trasferimento di un solo byte, poiché l'indirizzo è già stato specificato e l'incremento avviene automaticamente.

### La CPU legge il registro di stato del VDP

La CPU può leggere il registro di stato del VDP con il trasferimento di un solo byte. Viene attivata la linea MODE (alta) per il trasferimento e la linea CSR viene usata per indicare che si sta compiendo un'operazione di lettura.

### La CPU legge dalla VRAM

La lettura da parte della CPU nella VRAM avviene in modo del tutto analogo a quello della scrittura, anche in questo caso utilizzando il registro di indirizzamento ad autoincremento.

### Temporizzazione

Poiché la CPU agisce sulla VRAM tramite il VDP è ovvio che i trasferimenti possono avvenire solamente quando il VDP non è impegnato in at-

| OPERAZIONE | BIT 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $\overline{CSW}$ | $\overline{CSR}$ | MODO |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Scrittura in registri VDP | | | | | | | | | | | |
| Byte 1: Dato da scrivere | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 1 | 1 |
| Byte 2: Selezione del registro | 1 | 0 | 0 | 0 | 0 | RS2 | RS1 | RS0 | 0 | 1 | 1 |
| Scrittura nella VRAM | | | | | | | | | | | |
| Byte 1: Impostazione dell'indirizzo | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | 0 | 1 | 1 |
| Byte 2: Impostazione dell'indirizzo | 0 | 1 | A13 | A12 | A11 | A10 | A9 | A8 | 0 | 1 | 1 |
| Byte 3: Dato scritto | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 1 | 0 |
| Lettura registro VDP | | | | | | | | | | | |
| Byte 1: Dato letto | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 1 | 0 | 1 |
| Lettura della VRAM | | | | | | | | | | | |
| Byte 1: Impostazione dell'indirizzo | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | 0 | 1 | 1 |
| Byte 2: Impostazione dell'indirizzo | 0 | 0 | A13 | A12 | A11 | A10 | A9 | A8 | 0 | 1 | 1 |
| Byte 3: Dato letto | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 1 | 0 | 0 |

tività di visualizzazione di output sullo schermo; ne consegue che il tempo necessario per il trasferimento da parte della CPU di un byte da e per la VRAM varia da 2 a 8 microsecondi, in dipendenza dell'occupazione del VDP in attività di visualizzazione o "refresh" di memoria. Le temporizzazioni approssimative vengono indicate nella tabella 6.2.

## I registri del VDP

Solo i registri 0 e 1, tra gli otto a sola scrittura, contengono i flag che controllano la varie funzioni e modalità del VDP. I registri dal 2 al 6 contengono dei valori che indicano gli indirizzi di partenza dei vari sottoblocchi della VRAM, mentre il registro 7 viene usato per definire il colore di sfondo in tutte le modalità video e i colori del testo nella modalità a 40 colonne. Seguono ora le descrizioni dettagliate di questi registri:

### Registro 0

I due bit meno significativi sono bit di controllo. Tutti i rimanenti sono riservati per future espansioni e devono essere posti a zero.
Bit 0: Abilita/disabilita VDP esterno (1 = abilita)
Bit 1: Bit 3 di modalità: si veda il seguito per ulteriori dettagli.

### Registro 1 (8 bit di controllo del VDP)

Bit 7: Selezione 4/16K RAM (sempre a 1 nel sistema MSX)
Bit 6: Abilita/disabilita la cancellazione dello schermo (1 = abilita); la cancellazione implica che l'intero schermo assuma il colore di sfondo.
Bit 5: Abilita l'interruzione (1 = abilita, 0 = disabilita)
Bit 4: Bit 1 di modalità
Bit 3: Bit 2 di modalità; questi due ultimi, con il bit 3 del registro 0, definiscono la modalità di schermo, secondo lo schema riportato di seguito:

| *M1* | *M2* | *M3* | |
|---|---|---|---|
| *0* | *0* | *0* | *Modalità Grafica I* |
| *0* | *0* | *1* | *Modalità Grafica II* |
| *0* | *1* | *0* | *Modalità multicolore* |
| *1* | *0* | *0* | *Modalità testo* |

Bit 2: Riservato per future espansioni, deve essere posto a zero.
Bit 1: Selezione della dimensione degli sprite: 0 implica sprite di 8 * 8 pixel, 1 sprite di 16 * 16.

*Tab. 6.2 - Tempi di accesso alla VRAM*

| CONDIZIONE | MODALITÀ | RITARDO VDP | TEMPO D'ATTESA PER UNA FINESTRA D'ACCESSO | TEMPO TOTALE |
|---|---|---|---|---|
| Visualizzazione attiva | Testo | 2 μs | 0-1.1 μs | 2-3.1 μs |
| Visualizzazione attiva | Grafica I, II | 2 μs | 0-5.95 μs | 2-8 μs |
| Visualizzazione attiva | Multicolore | 2 μs | 0-1.5 μs | 2-3.5 μs |
| Bit "blank" del registro 1=0 | Tutte | 2 μs | 0 μs | 2 μs |
| 4300 μs dopo l'interruzione | Tutte | 2 μs | 0 μs | 2 μs |

Bit 0: Selezione dell'opzione d'ingrandimento degli sprite: 0 implica sprite di dimensioni normali, 1 sprite ingranditi.

**Registro 2**

I quattro bit meno significativi del registro 2 formano i quattro bit di ordine più alto dell'indirizzo, su quattordici bit, della tabella dei nomi; ne consegue che l'indirizzo di partenza della tabella dei nomi è pari a (Registro 2)*400H.

**Registro 3**

Il registro 3 determina l'indirizzo della VRAM da cui parte la tabella dei colori. Il contenuto di questo registro costituice gli otto bit di più alto ordine dell'indirizzo su 14 bit: quindi quest'ultimo può essere calcolato come (Registro 3)*40H.

**Registro 4**

I tre bit meno significativi definiscono l'inizio del generatore di sottoblocchi per le tabelle dei modelli, dei testi e multicolore, formando così i tre bit piu significativi dell'indirizzo. Ne consegue che l'indirizzo del generatore di sottoblocchi deve essere calcolato come (Registro 4)*800H.

**Registro 5**

I sette bit più "bassi" di questo registro formano i 7 bit più significativi dell'indirizzo della tabella degli attributi degli sprite; quindi l'indirizzo di base è dato da (Registro 5)*80H.

**Registro 6**

Il registro 6 definisce l'indirizzo d'inizio, nella VRAM, della tabella generatrice dei modelli degli sprite. L'indirizzo è uguale a (Registro 6)*800H.

**Registro 7**

I quattro bit più significativi del registro 7 contengono il codice di colore per il colore nella modalità testo, mentre i quattro bit meno significativi determinano il codice di colore per il colore 0 nella modalità testo, o il colore dello sfondo per tutte le altre modalità.

**Il registro di stato**

Il VDP ha un solo registro di stato su 8 bit che può essere accessibile dalla CPU. Questo registro contiene i flag d'interruzione, di collisione tra due

sprite, di "quinto sprite", e il numero del quinto sprite. Il formato del registro di stato viene descritto in seguito.
Il registro di stato può essere letto in qualsiasi momento. Però se esso viene letto in modo asincrono rispetto all'interruzione generata dal segnale video, il flag connesso alla memoria video viene posto a zero e ciò può causare la perdita dell'interruzione stessa. Per evitare che ciò accada è consigliabile leggere la copia del registro di stato del VDP che viene mantenuta dal sistema operativo nella RAM (si veda Capitolo 5).

**Bit 7: il flag d'interruzione**

Questo flag viene posto a 1 alla fine della scansione elettronica dell'ultima linea del video. La conseguenza è quella di attivare anche la linea d'interruzione, a patto che sia posto a 1 il flag di abilitazione delle interruzioni del registro 0. Il flag d'interruzione viene rimesso a zero quando si legge il registro di stato; si noti che questa lettura deve essere effettuata al termine di ogni scansione video proprio allo scopo di rinizializzare il flag di interruzione e quindi riabilitarlo per la successiva scansione video.

**Bit 5: il flag di sovrapposizione di sprite**

Questo flag viene posto a 1 quando si verifica una collisione tra due sprite (cioè quando si hanno uno o più pixel in sovrapposizione). Gli sprite trasparenti vengono considerati, così come quelli che sono totalmente o parzialmente fuori dallo schermo. Gli sprite che si trovano oltre il delimitatore della tabella degli attributi degli sprite (0D0H) non vengono però considerati. Il flag viene rimpostato a zero quando viene letto il registro di stato.

**Bit 6: il flag di "quinto sprite"**

Il bit 6 del registro di stato viene posto a 1 quando su una medesima linea dello schermo sono presenti cinque o più sprite contemporaneamente. Viene posto a zero quando viene letto il registro di stato. Quando è posto a 1, il numero del quinto sprite viene messo nei cinque bit meno significativi del registro di stato, consentendo all'utente di muovere lo sprite prima della prossima scansione video, in modo che tutti gli sprite siano posizionati correttamente.

## Modalità di visualizzazione

Il VDP genera un'immagine che può essere pensata come composta da un certo numero di piani sovrapposti. L'ordine di priorità di questi piani è legato all'ordine di sovrapposizione in profondità (il piano più lontano è quello

a più bassa priorità): ne consegue che se due oggetti si trovano nella medesima posizione su due piani diversi, l'oggetto che si trova sul piano a più alta priorità precluderà l'altro (e ogni altro che si trovi su piani di priorità inferiore). Se ne può quindi dedurre che uno sprite sul piano 0 apparirà davanti a tutto ciò che compare nella schermata attuale. Partendo dal più esterno, questi piani sono: i 32 piani per gli sprite, seguiti dal piano dei caratteri, seguito dal piano di sfondo, seguiti a loro volta dai piani relativi agli eventuali VDP connessi esternamente. Non sembra tuttavia che le compagnie che producono lo standard MSX intendano usare più di un processore video.
Il piano dello sfondo presenta una colorazione uniforme che viene usata per la visualizzazione delle aree del bordo ed è il colore di default per il piano dei caratteri.
Quando viene impostato con il colore trasparente, automaticamente assume per default il nero, a meno che non siano presenti altri VDP esterni. I 32 piani degli sprite sono posti davanti al piano dei caratteri: lo sprite 31 ha la priorità più bassa e lo sprite 0 quella più alta. Questi piani vengono disattivati nella modalità testo. Per default gli sprite sono trasparenti e le loro posizioni possono essere definite pixel per pixel, consentendo quindi un movimento senza scatti. Il contenuto e la memorizzazione del piano dei caratteri varia tra le modalità di visualizzazione e quindi verrà discusso separatamente.
In tutte le modalità sono disponibili i seguenti 16 colori:

| CODICE | COLORE |
|---|---|
| 0 | Trasparente |
| 1 | Nero |
| 2 | Verde medio |
| 3 | Verde chiaro |
| 4 | Blu scuro |
| 5 | Blu chiaro |
| 6 | Rosso scuro |
| 7 | Ciano |
| 8 | Rosso medio |
| 9 | Rosso chiaro |
| 10 | Giallo scuro |
| 11 | Giallo chiaro |
| 12 | Verde scuro |
| 13 | Rosso Magenta |
| 14 | Grigio |
| 15 | Bianco |

## Modalità Grafica I

Nella modalità grafica I il sottoblocco di VRAM che costituisce la tabella dei nomi è composto da 768 byte, logicamente divisi in 24 righe per 32 caratteri. Quindi i primi 32 caratteri della tabella rappresentano la prima linea del video, i successivi 32 la seconda e così via. Ciascun elemento della tabella fa riferimento a una delle 256 definizioni di modello contenute nella tabella di generazione dei modelli. Ognuna di queste definizioni è composta da otto byte (8 * 8 bit), e quindi l'occupazione totale è di 2048 byte della VRAM. In queste definizioni un bit posto a 1 implica la visualizzazione del pixel corrispondente con colore 1; un bit posto a 0 implica il colore 0. I colori per ciascun modello vengono determinati dalla tabella dei colori (che nella modalità Grafica I è lunga 32 byte). In questa tabella ogni elemento definisce un colore di primo piano (colore 1) e uno di sfondo (colore 0) per otto modelli della tabella di generazione dei modelli. Infatti il primo elemento della tabella dei colori si riferisce ai modelli dallo 0 al 7, il secondo ai modelli dall'8 al 15 e così via. Gli elementi della tabella dei colori sono logicamente considerati come due semibyte: il più significativo indica il colore di primo piano, l'altro il colore di sfondo. Da quanto detto si può dedurre che la completa implementazione di questa modalità richiede l'occupazione di 2848 byte di VRAM: se però non è necessaria la definizione di tutti i 256 modelli, è possibile sovrapporre parzialmente le tabelle.

## Modalità Grafica II

Nella modalità Grafica II, la tabella dei nomi viene rappresentata come nella modalità Grafica I, con l'eccezione che ogni elemento di questa tabella può avere la sua unica definizione nella tabella di generazione dei modelli.
Ciò è possibile in quanto la tabella di generazione dei modelli viene divisa in tre blocchi, ciascuno comprendente 256 definizioni; i primi 256 elementi della tabella dei nomi (vale a dire la parte superiore dello schermo) si riferiscono alle prime 256 definizioni della tabella di generazione dei modelli; analogamente gli elementi della parte centrale fanno riferimento alle definizioni dalla 256 alla 511 e quelli della parte inferiore dello schermo usano le definizioni dalla 512 alla 767.
È inoltre possibile definire il colore di primo piano e di sfondo di ogni linea orizzontale di ciascuna definizione di modello. Infatti la tabella dei colori è formata da 768 elementi, ciascuno di otto byte. Ciascuno di questi è in corrispondenza con una definizione della tabella di generazione dei modelli. I quattro bit più significativi di ogni byte della tabella dei colori definiscono il colore di primo piano del byte corrispondente nella tabella dei modelli, mentre gli altri quattro determinano il colore di sfondo.

*Fig. 6.1 - Mappa dello schermo "bit-mapped" nella modalità Grafica II.*

Come si può vedere, se si usa un'attenta configurazione dei valori contenuti nella tabella dei nomi, la modalità Grafica II può essere considerata "bit-mapped"; viceversa copiando i medesimi valori in tutti e tre i blocchi della tabella di generazione dei modelli e nella tabella dei colori questa modalità può essere sfruttata in modo analogo alla modalità Grafica I, ma con una migliore definizione del colore.

### Modalità multicolore

La modalità multicolore consente di utilizzare una schermo con 64 * 48 quadrati di colore, ciascuno dei quali è formato da un blocco di 4 * 4 pixel. Ognuno di questi quadrati può essere in uno dei 15 colori disponibili nel VDP.
La tabella dei nomi nella modalità multicolore è mappata in modo simile alle due modalità grafiche, anche se non vi è più riferimento ad una tabella di colori. Infatti il colore viene ora determinato dagli elementi della tabella di generazione dei modelli.
Ciascun valore nella tabella dei nomi punta ad un blocco di otto byte nella tabella di generazione dei modelli. Solo due di questi byte vengono usati per definire l'immagine che viene visualizzata: vengono considerati come una serie di quattro semibyte che definiscono un modello multicolore di 8 * 8 pixel. Il semibyte di ordine più alto del primo byte definisce il colore del quadrato di 4 * 4 pixel che si trova in alto sulla sinistra nel modello di 8 * 8 pixel; il semibyte meno significativo definisce il colore del quadrato in alto a destra; il secondo byte si comporta in modo analogo per quanto riguarda i quadrati in basso, rispettivamente a sinistra e a destra, nel modello multicolore di 8 * 8 pixel. Gli elementi delle righe 0, 4, 8, 12, 16 e 20 della tabella dei nomi usano i primi due byte del blocco di otto byte della tabella di generazione dei modelli, mentre gli elementi delle righe 1, 5, 9, 13, 17 e 21 usano i successivi due byte e così via.

### Modalità testo

Nella modalità testo lo schermo viene considerato come una griglia di 40 caratteri su 24 linee. Ciascun elemento della griglia si estende per 6 pixel in orizzontale e per 8 in verticale. Le tabelle che vengono utilizzate per generare il piano dei modelli sono la tabella dei nomi e la tabella di generazione dei modelli: possono esistere contemporaneamente 256 definizioni di modelli. La tabella dei nomi collega queste definizioni con ciascuno dei 960 elementi del piano dei modelli. In questa modalità non sono disponi-

bili gli sprite. La tabella di generazione dei modelli è identica a quella della modalità Grafica I con l'eccezione che, poiché ogni carattere si estende in orizzontale per soli 6 pixel, i due bit meno significativi di ciascun byte nella tabella di generazione dei modelli vengono ignorati. I colori di primo piano e di sfondo sono determinati dal valore del registro 7 del VDP: il semibyte alto definisce quello di primo piano, mentre quello basso determina il colore di sfondo e del bordo.

**Sprite**

Sullo schermo possono essere visualizzati fino a 32 sprite, che si muovono sui piani a più alta priorità. La loro posizione viene definita dall'angolo in alto a sinistra del modello di definizione, e, poiché la posizione può essere definita pixel per pixel, risulta semplice far muovere lentamente e senza scatti lo sprite sul suo piano. Ciascuno dei piani degli sprite è del tutto trasparente tranne la zona in cui si trova lo sprite stesso. Gli sprite non sono disponibili nella modalità testo a 40 colonne.
Nella VRAM si trovano la tabella degli attributi degli sprite e la tabella di generazione dei modelli degli sprite. Queste tabelle sono sostanzialmente analoghe a quelle che agiscono nel piano dei modelli: infatti la tabella degli attributi degli sprite determina la posizione dello sprite, mentre la tabella dei modelli degli sprite contiene la definizione del suo disegno.
Ciascuno dei 32 elementi della tabella degli attributi degli sprite consiste in quattro byte: questi indicano la sua coordinata verticale, seguita da quella orizzontale, seguita dal nome (che fa riferimento alla tabella di generazione dei modelli per gli sprite), seguita infine dal suo "tag", i cui quattro bit meno significativi ne definiscono il colore e il bit più significativo è il bit di "early clock". Quest'ultimo fa in modo che lo sprite appaia 32 pixel sulla sinistra della sua posizione, che, come detto, è determinata dai primi due byte degli elementi della tabella degli attributi degli sprite.
Quando uno sprite viene posizionato in 0,0 appare attaccato al margine sinistro in alto del bordo. In molte applicazioni però è necessario che lo sprite faccia la sua apparizione sullo schermo partendo dai bordi estremi. Per ottenere questo effetto esiste un metodo differente per ognuno dei due assi, vale a dire:-
Innanzitutto la posizione verticale può assumere anche valori negativi: i valori compresi tra —31 e 0 (in complemento a due) permettono allo sprite di apparire gradualmente dal margine superiore dello schermo. Si può però facilmente dedurre che, poiché la risoluzione orizzontale dello schermo è di 256 pixel, questo metodo non può essere applicato al margine sinistro; è questo appunto lo scopo per il quale è stato introdotto il bit di "early

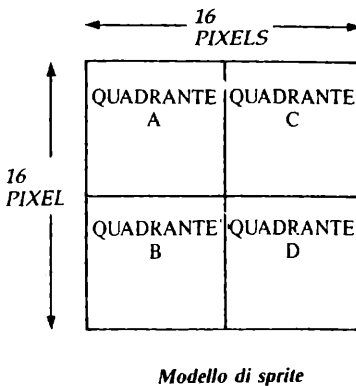
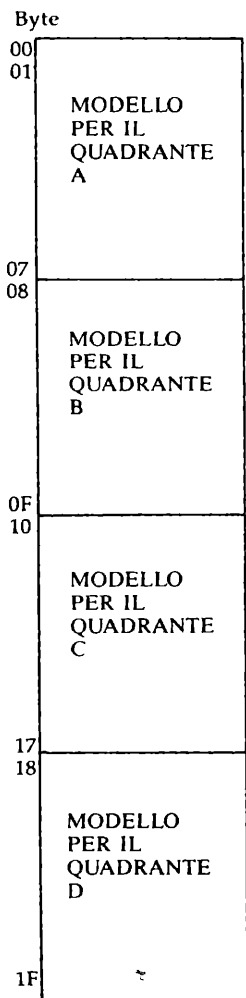



*Fig. 6.2 - Mappa per gli sprite 16*16.*

clock" nel quarto byte di ogni elemento della tabella degli attributi degli sprite.
La tabella di generazione dei modelli per gli sprite consiste in un massimo di 256 elementi di otto byte; il terzo byte di ciascun elemento della tabella degli attributi degli sprite definisce quale elemento della tabella di generazione dei modelli debba essere usato per lo sprite nel piano che si sta considerando. Si noti che quando si usano sprite di 16 * 16 pixel questo valore è considerato per multipli di quattro, puntando quindi a blocchi di 32 byte nella tabella di generazione degli sprite.
Si deve infine notare che su una linea orizzontale possono essere visualizzati fino ad un massimo di quattro sprite. Nel caso in cui questa regola venisse violata, appaiono sullo schermo solo i quattro sprite a più alta priorità; inoltre il flag di "quinto sprite" nel registro di stato del VDP verrebbe posto a 1, e il numero del quinto sprite verrebbe memorizzato nei suoi cinque bit di più basso ordine.

## Il VDP in ambiente MSX

Nel sistema MSX il VDP può essere accessibile in due modi. In primo luogo la ROM di sistema del MSX fornisce una serie di procedure con le quali l'utente può effettuare la maggior parte delle operazioni necessarie. In qualche caso però queste procedure possono rivelarsi inadeguate per la funzionalità offerta o per il tempo di esecuzione: in tal caso è possibile un accesso diretto al VDP tramite le locazioni 6 e 7 della ROM di sistema, in cui sono contenuti gli indirizzi di scrittura e lettura per il VDP.
Può forse apparire strano, a prima vista, che siano necessari due indirizzi: un'analisi più approfondità ci porterà a capire che la giustificazione di ciò è la linea di controllo MODE del VDP.
Facendo riferimento alla tabella 6.1, possiamo vedere che per scrivere in un registro del VDP la linea di controllo MODE deve essere attiva (alta), mentre deve essere disattiva (bassa) per una lettura o scrittura nella VRAM. Poiché, come già detto in precedenza, la linea di controllo MODE è legata a una delle linee d'indirizzo della CPU, potremmo aspettarci che gli indirizzi memorizzati nelle locazioni 6 e 7 siano tra loro differenti per un solo bit. Esaminando la ROM, troviamo che la locazione 6 contiene 98H e la 7 contiene 99H: se ne può dedurre che la linea MODE è legata alla linea 0 d'indirizzo. Le altre due linee di controllo del VDP, la CSR e la CSW sono legate alle linee di controllo IORQ, RD e WR della CPU, in modo che il loro controllo venga effettuato automaticamente. La procedura riportata qui di seguito scrive, nella locazione di VRAM specificata da un indirizzo su 14 bit contenuto nella coppia di registri HL, il dato contenuto

nell'accumulatore, ed è utile per illustrare quanto appena detto:

```
WRTVRM:   PUSH AF      ;SALVA I DATI
          LD A,(6)     ;PRENDE L'INDIRIZZO DI VDP CON LA
                       ;LINEA MODE ALTA
          LD C,A       ;LO COPIA IN C
          INC C        ;LINEA MODE ALTA
          DI           ;DISABILITA LE INTERRUZIONI
          OUT (C),L    ;USCITA DEL BYTE INFERIORE
                       ;DELL'INDIRIZZO
          SET 6,H      ;PONE A 1 IL BIT 6 DEL BYTE SUPERIORE
                       ;DELL'INDIRIZZO
          OUT (C),H    ;LO FA USCIRE
          DEC C        ;LINEA MODE BASSA
          POP AF       ;RIPRISTINA I DATI
          OUT (C),A    ;LI STAMPA
          EI           ;RIABILITA LE INTERRUZIONI
          RET          ;RETURN
```

Il sistema operativo rende disponibili delle routine, ben scritte e ottimizzate come tempo di esecuzione, per tutte le funzioni che è probabile siano necessarie; le prenderemo ora in esame:

| INDIRIZZO | FUNZIONE |
|---|---|
| 0041H | Disabilita la visualizzazione su schermo. Non è richiesto alcun parametro, ma vengono modificate le coppie di registri AF e BC. |
| 0044H | Abilita la visualizzazione su schermo: anch'essa non richiede alcun parametro e modifica le coppie AF e BC. |
| 0047H | Scrive il dato posto in B nel registro specificato da C, modificando le coppie AF e BC. |
| 004AH | Legge un byte della VRAM all'indirizzo contenuto nella coppia HL: il valore viene ritornato nell'accumulatore e viene modificata solo la coppia AF. |
| 004DH | Scrive il dato contenuto nell'accumulatore all'indirizzo di VRAM specificato dalla coppia HL, modificando solamente AF. |
| 0050H | Prepara il VDP per un'operazione di lettura dalla VRAM: accetta come parametro l'indirizzo di partenza in HL e modifica la coppia AF. |
| 0053H | Analoga alla precedente per quanto concerne un'operazione di scrittura. |

| | |
|---|---|
| 0056H | Questa routine riempie il numero di byte di VRAM specificati nella coppia BC, a partire dall'indirizzo posto in HL con i dati passati per mezzo dell'accumulatore. Modifica le coppie AF e BC. |
| 0059H | Sposta un blocco di memoria dalla VRAM alla RAM. È analoga alla routine che segue. |
| 005CH | Sposta un blocco di memoria dalla memoria della CPU alla VRAM. L'indirizzo di partenza è in HL, quello di destinazione in DE e la lunghezza del blocco è posta in BC; tutti i registri vengono modificati. |
| 005FH | Imposta il VDP alla modalità di visualizzazione definita dalla locazione 00CAFH (0 per testo 40, 1 per Grafica I, 2 per grafica II, 3 per multicolore). Modifica tutti i registri. |
| 0062H | Cambia il colore dello schermo: i parametri sono il colore di primo piano (nella locazione 0F3EAH), quello di sfondo (locazione 0F3EAH) e quello del bordo (locazione 0F3EBH). Modifica tutti i registri. |
| 0069H | Inizializza tutti gli sprite. I modelli degli sprite vengono cancellati, i nomi degli sprite vengono posti uguali ai numeri del piano di ogni sprite, i colori sono posti uguali al colore di primo piano e le posizioni verticali al valore 209. Questa routine richiede che venga passata la modalità video nella locazione 0FCAFH e modifica inoltre tutti i registri. |
| 006CH | Inizializza il VDP per la modalità testo su 40 colonne. Questa routine usa i valori che si aspetta di trovare nell'area della RAM di sistema dall'indirizzo 0F3B3H in avanti e modifica tutti i registri. |
| 006FH | Inizializza il VDP per la modalità Grafica I; le due routine che seguono sono del tutto analoghe a questa. |
| 0072H | Inizializza il VDP per la modalità Grafica II. |
| 0075H | Inizializza il VDP per la modalità multicolore. |

In circostanze normali è piuttosto improbabile che si renda necessario un accesso diretto al VDP da parte dell'utente.

## La programmazione del VDP: informazioni e suggerimenti

### Il piano di visualizzazione dei caratteri

Nelle modalità Grafica I e II, che sono quelle che più interessano il programmatore, quel che appare sul piano dei caratteri è completamente de-

terminato dalle sue scelte: infatti la definizione di tutti i caratteri viene presa dalla RAM. Si può però facilmente dedurre che sarebbe consigliabile, nella definizione di un proprio insieme di caratteri, seguire certe indicazioni di massima. Per esempio, lavorando sulle macchine MSX si è visto che è consigliabile mettere almeno la definizione delle lettere maiuscole e dei numeri in locazioni della VRAM in modo che sia possibile scrivere il codice ASCII di una lettera o di una cifra nella tabella dei nomi per ottenerne la visualizzazione. Per esempio la definizione di una 'A' (codice ASCII 65) dovrebbe essere posta a un indirizzo della VRAM calcolabile nel seguente modo: (indirizzo di base dalla tabella di generazione dei modelli) + (65 * 8). Per esperienza personale ho potuto constatare che il modo più semplice per ottenere ciò consiste nel prendere le definizioni di base dalla ROM e ridefinire quindi quelle che si vuole. Quest'operazione viene fatta dal sistema operativo nella modalità Grafica I, con un comando SCREEN 1; però quando si ritorna alla modalità Grafica II con un comando SCREEN, il sistema operativo imposta il VDP con tutte le 768 definizioni rinizializzate. Diventa quindi necessario localizzare nella ROM di sistema MSX la definizione standard dei caratteri. Essa può cambiare da macchina a macchina: il piccolo programma che segue serve per individuare l'indirizzo della 'A' nella ROM MSX:

```
10 FOR N% =&H0000 TO &H7FFF
20 A% =PEEK(N%):READ B%
30 IF A% <> B% THEN RESTORE:Q% =0 ELSE Q% =Q% + 1
40 IF Q% =8 THEN PRINT HEX$(N%—8):END
50 NEXT N%
60 END
70 DATA 32,80,136,136,248,136,136,0
```

I programmi BASIC e in codice macchina che seguono illustrano molte delle questioni che devono essere prese in considerazione quando si programma il VDP. Viene inoltre fornito un potente programma per la definizione di caratteri e di sprite, principalmente da usare nella modalità Grafica II (ma funziona anche nelle altre modalità).

```
WRTVDP EQU 47H    ;CHIAMA LA TABELLA DEI BIOS
RDVRM  EQU 4AH
WRTVRM EQU 4DH
FILVRM EQU 56H
LDIRVM EQU 5CH
LDIRMV EQU 59H
CHGET  EQU 9FH
```

```
        GTSTCK  EQU 0D5H
        GTTRIG  EQU 0D8H
        RDVDP   EQU 13EH
        SNSMAT  EQU 141H    ;FINE DELLA TABELLA DEI BIOS

***** MODO GRAFICA II DEFINITORE DI COLORI/CARATTERI *****
        NAMTAB EQU 1800H
        COLTAB  EQU 2000H ;INDIRIZZI DI VRAM PER GM2
        PATTAB  EQU 0000H
        SATTAB  EQU 1B00H
        SPTTAB  EQU 3800H ;FINE INDIRIZZI DI VRAM
        RNAMTB EQU 06H     ;VALORI DEL REGISTRO VDP
        RCOLTB  EQU 0FFH   ;PER MODO GRAFICA II
        RPTTAB  EQU 03H
        RSATAB  EQU 36H
        RSPTAB  EQU 07H    ;FINE VALORI DEL REGISTRO
        CHARDT  EQU 1CBFH ;L'ALLOCAZIONE DEI CARATTERI
                           ;NELLA ROM PUÒ ESSERE
                           ;DIVERSA
        ORG 9000H

REGDAT: DEFB 2,0E0H,RNAMTB,RCOLTB,RPTTAB,RSATAB,RSPTAB,6

START:  LD E,8
        LD IX,REGDAT          ;INDICATORE DEI DATI IN IX
        LD D,0                ;CARICA I REGISTRI
SVDPLP: LD B,(IX+0)           ;DATI IN B
        LD C,D                ;NUMERO DEL REGISTRO IN C
        CALL WRTVDP           ;AL VDP
        INC IX                ;INCREMENTA L'INDICATORE
                              ;DEI DATI
        INC D                 ;INCREMENTA IL NUMERO
                              ;DEL REGISTRO
        DEC E
        JR NZ,SVDPLP          ;FORMA 8 REGISTRI
        LD HHL,CHARDT         ;PRENDE L'INDIRIZZO
                              ;DEI CARATTERI
        LD DE,PATTAB+1100H ;DE È L'INDIRIZZO DI VRAM
        LD BC,60*8            ;FORMA 60 CARATTERI
        CALL LDIRVM
        LD BC,800H
        LD HL,COLTAB+1000H
```

```
              LD A,0F1H          ;FORMA I COLORI
              CALL FILVRM
              LD BC,800H
              LD HL,COLTAB
              LD A,1FH
              CALL FILVRM        ;FORMA I COLORI
              LD BC,800H
              LD HL,COLTAB+800H
              LD A,81H
              CALL FILVRM        ;PIÙ COLORI
              LD BC,768
              LD HL,NAMTAB
              LD A,32
              CALL FILVRM        ;PULISCE LO SCHERMO
              LD HL,P2DAT
              LD DE,PATTAB+800H
              LD BC,24
              CALL LDIRVM        ;DEFINISCE I CARATTERI
                                 ;DA 0 A 2 NEL BLOCCO 2
              LD HL,P2DAT+16
              LD DE,SPTTAB
              LD BC,8
              CALL LDIRVM        ;DEFINISCE SPRITE 0
              CALL FUNCPT        ;STAMPA LE COLONNE
                                 ;DELLA FUNZIONE
              CALL FTTXRM        ;TRASFERISCE I DATI VRAM
                                 ;ALLA RAM
MLP:          CALL ARRYPT        ;STAMPA LA MATRICE
                                 ;DI DEFINIZIONE
              CALL SPON          ;POSIZIONA IL CURSORE
              CALL JOY           ;LEGGE IL JOYSTICK
              CALL FUNC          ;STAMPA I MESSAGGI
                                 ;DELLA FUNZIONE
              HALT               ;ASPETTA IL FLYBACK
              CALL DOFUNC        ;ESEGUE LE FUNZIONI
NDFNC:        HALT               ;ASPETTA IL FLYBACK
              HALT
              CALL CHART         ;TRASFERISCE LA DEFINIZIONE
              CALL CSPT          ;DALLA MATRICE ALLA RAM
              CALL COLPT         ;E STAMPA MESSAGGI
                                 ;DI STATUS
              CALL BANKPT
```

```
        CALL LPTXRM          ;TRASFERISCE I DATI VRAM
                             ;ALLA RAM
        HALT                 ;ASPETTA IL FLYBACK
        JR MLP               ;TORNA INDIETRO E RIPETE
        RET
SPY:    DEFB 0               ;POSIZIONE DEL CURSORE Y
SPX:    DEFB 0               ;POSIZIONE DEL CURSORE X
ARRYPT: LD C,8               ;ROUTINE CHE STAMPA
                             ;LA MATRICE DI DEFINIZIONE
        LD HL,NAMTAB+259     ;INDIRIZZO DELLA TABELLA
                             ;NOMI+OFFSET
        LD IX,HARRY1         ;INDIRIZZO DEL VETTORE IN IX
APTOLP: LD B,8               ;LOOP INTERNO DI 8
APTILP: LD A,(IX+0)          ;PRENDE I DATI DEL VETTORE
        PUSH BC              ;SALVA I CONTATORI
        CALL WRTVRM          ;SCRIVE I DATI DEL VETTORE
                             ;SULLO SCHERMO
        POP BC               ;RIPRISTINA I CONTATORI
        INC HL               ;INCREMENTA IL PUNTATORE
                             ;DELLO SCHERMO
        INC IX               ;INCREMENTA IL PUNTATORE
                             ;AL VETTORE
        DJNZ APTILP          ;FINE DEL LOOP INTERNO
        LD DE,24             ;LUNGHEZZA DI LINEA 8
        ADD HL,DE            ;SOMMATA AL PUNTATORE
                             ;ALLO SCHERMO
        DEC C                ;DECREMENTA IL CONTATORE
                             ;DEL LOOP ESTERNO
        JR NZ,APTOLP         ;RIPETE 8 VOLTE
        RET
FUNCPT: LD HL,NAMTAB+258     ;INDIRIZZO DELLA COLONNA
                             ;SINISTRA DEL VETTORE
        CALL ROWOF8          ;STAMPA LA COLONNA
        LD HL,NAMTAB+267     ;DESTRA DEL VETTORE
        CALL ROWOF8          ;STAMPA LA COLONNA
        RET
ROWOF8: LD B,8               ;CONTEGGIO DI 8
RO8LP:  PUSH BC              ;SALVA IL CONTATORE
        PUSH HL              ;SALVA L'INDIRIZZO
        LD A,2               ;CARATTERE 2
        CALL WRTVRM          ;ALL'INDIRIZZO
                             ;DELLO SCHERMO
```

```
        POP HL               ;RIPRISTINA L'INDIRIZZO
                             ;DELLO SCHERMO
        LD DE,32             ;SOMMA LA LUNGHEZZA
                             ;DELLA LINEA...
        ADD HL,DE            ;ALL'INDIRIZZO
                             ;DELLO SCHERMO
        POP BC               ;RIPRISTINA IL CONTATORE
        DJNZ R08LP           ;FA IL CONTEGGIO DI 8
        RET
SPON:   LD HL,SATTAB         ;ROUTINE DI POSIZIONAMENTO
                             ;DEL CURSORE
        LD A,(SPY)           ;CURSORE DELLA COORDINATA Y
        SLA A
        SLA A                ;MOLTIPLICA PER 8
        SLA A                ;POICHÉ LA POSIZIONE
                             ;DI SPRITE È CHARPOS*8
        ADD A,63             ;APICE SINISTRO
                             ;DELLA MATRICE, OFFSET
        CALL WRTVRM          ;SCRIVE NELLA TABELLA
                             ;ATTRIBUTI SPRITE
        INC HL               ;INCREMENTA IL PUNTATORE
                             ;DI VRAM
        LD A,(SPX)           ;PRENDE LO SPRITE X
        SLA A
        SLA A
        SLA A                ;*8
        ADD A,16             ;SOMMA OFFSET DI X
        CALL WRTVRM          ;ALLA TAVOLA DEGLI ATTRIBUTI
                             ;DI SPRITE
        INC HL               ;PUNTA AL NOME DI SPRITE
        LD A,0               ;IL NOME È 0
        CALL WRTVRM          ;SCRIVE GLI ATTRIBUTI
                             ;DI SPRITE
        INC HL               ;INCREMENTA IL PUNTATORE
        LD A,15              ;SPRITE È BIANCO
        CALL WRTVRM          ;AGLI ATTRIBUTI
        RET
P2DAT:  DEFB 255,129,129,129,129,129,129,255 ;DEFINIZIONI...
        DEFB 255,255,255,255,255,255,255,255 ;DEI CARATTERI
        DEFB 255,195,165,153,153,165,195,255 ;0—2 E SPRITE 0
HARRY1: DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0
```

```
        DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0   ;VETTORE DEI CARATTERI
        DEFB 0,0,0,0,0,0,0,0
HARRY2: DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0
        DEFB 0,0,0,0,0,0,0,0   ;VETTORE TEMPORANEO
        DEFB 0,0,0,0,0,0,0,0   ;PER TRASFORMAZIONI
JOY:    LD A,0                 ;PER LEGGERE JOY(0)
                               ;O LE FRECCE DEL CURSORE...
        CALL GTSTCK            ;CAMBIANO A PIACERE
                               ;PER LEGGERE I JOYSTICKS
        LD HL,(SPY)            ;CURSORI X,Y IN HL
        CP1                    ;JOYSTICK AVANTI?
        CALL Z,DEY             ;SE SÌ DECREMENTA Y
        CP3                    ;JOYSTICK A DESTRA?
        CALL Z,INX             ;SE SÌ INCREMENTA X
        CP5                    ;JOYSTICK INDIETRO?
        CALL Z,INY             ;SE SÌ INCREMENTA Y
        CP7                    ;JOYSTICK A SINISTRA?
        CALL Z,DEX             ;SE SÌ DECREMENTA X
        LD A,L                 ;DA Y
        CP8
        JR NZ,JOY1
        LD L,0                 ;SE NECESSARIO WRAP AROUND
JOY1:   CP 255
        JR NZ,JOY2
        LD L,7
JOY2:   LD A,H                 ;ULTERIORE WRAP AROUND
        CP10
        JR NZ,JOY3
        LD H,0                 ;SE NECESSARIO WRAP
                               ;AROUND SU X
JOY3:   CP 255
        JR NZ,JOY4
        LD H,9
```

```
JOY4:     LD (SPY),HL        ;MEMORIZZA X E Y
                             ;DEL CURSORE
          RET
INY:      INC L
          RET
DEY:      DEC L
          RET
INX:      INC H
          RET
DEX:      DEC H
          RET
EXIT:     LD A,0             ;USCITA DALLA ROUTINE
          LD (BASFNC),A      ;L'USCITA È LA FUNZIONE 0
          POP BC             ;RIMUOVE L'INDIRIZZO
                             ;DI RITORNO
          RET                ;RITORNA AL BASIC
GETMGS:   LD A,(MENÙ)        ;MENÙ 0 O 1
          LD IY,MGSTAB       ;IY PUNTA AI MESSAGGI
          SLA A              ;MENÙ DIVENTA 0 O 2
          LD C,A
          LD B,0
          ADD IY,BC          ;IY È IY O IY+2
          LD A,(IY+0)        ;ORDINE INFERIORE
                             ;DELL'INDIRIZZO DEI MESSAGGI
          LD C,A             ;IN C
          LD A,(IY+1)        ;ORDINE SUPERIORE
          LD B,A             ;IN B
          PUSH BC            ;PRENDE L'INDIRIZZO
                             ;DELLA TABELLA DEI MESSAGGI
          POP IX             ;IN IX
          RET
DFUNC:    DEFB 0
MGSTAB:   DEFW FNMSGS,       ;CONSERVA GLI INDIRIZZI
          FNMSG1             ;DELLE TAVOLE DEI MESSAGGI
MENU:     DEFB 0             ;NUMERO DI MENÙ
FUNC:     CALL SPPNT         ;CANCELLA IL MESSAGGIO
                             ;DELLA VECCHIA FUNZIONE
          LD A,(SPX)         ;PRENDE LO SPRITE X
          CP 0               ;COLONNA DI FUNZIONE
                             ;DESTRA?
          JR Z,FUNC1         ;SE È COSÌ STAMPA IL
                             ;MESSAGGIO DELLA FUNZIONE
```

```
        CP 9                    ;COLONNA SINISTRA?
        RET NZ                  ;SE NO RETURN
        LD A,8
FUNC1:  LD B,A
        LD A,(SPY)
        ADD A,B
        LD B,A
        LD (DFUNC),A            ;CALCOLA E MEMORIZZA
                                ;IL NUMERO DELLA FUNZIONE
        PUSH BC                 ;SALVA BC
        CALL GETMGS             ;PRENDE L'INDIRIZZO
                                ;DEL MESSAGGIO
        POP BC                  ;RIPRISTINA BC
        CALL MSGFND             ;TROVA IL MESSAGGIO
                                ;RELATIVO NELLA TABELLA
FPNT:   LD HL,NAMTAB+515
FNPNT1: LD A,(IX+0)
        CP '$'
        RET Z
        CALL WRTVRM
        INC HL
        INC IX
        JR FNPNT1               ;STAMPA IL MESSAGGIO
SPPNT:  PUSH AF                 ;ROUTINE STAMPA SPAZI
        LD IX,NOFUNC
        CALL FNPNT
        POP AF
        RET
MSGFND: DEC B                   ;TROVA IL MESSAGGIO B
                                ;NELLA STRINGA...
        RET M
MSGFN1: LD,(IX+0)               ;INDIRIZZATO DA IX
        INC IX
        CP '$'
        JR NZ,MSGFN1
        JR MSGFND
NOFUNC: DEFB '               $'     ;STRINGA FITTIZIA IN
                                    ;ASSENZA DI FUNZIONE
FNMSGS: DEFB 'EXIT$LOAD$SAVE$CLEAR$';MENÙ 1 STRINGA
                                          ;DI MESSAGGIO
        DEFB 'CHARACTER$SPRITE$COLOUR$'
        DEFB 'SCROLL LEFT$SCROLL RIGHT$'
```

```
        DEFB 'SCROLL UP$SCROLL DOWN$'
        DEFB 'COPY$INVERT$FLIP VERTICAL$'
        DEFB 'FLIP HORIZONTAL$MENU 2$'
FNMSG1: DEFB 'POSITION$SWAP BANK$MAGNIFY SPRITE$'
                                        ;MENÙ 2 STRINGA
        DEFB 'DEMAGNIFY SPRITE$16*16 SPRITE$'
        DEFB '8*8 SPRITE$NOT IMPLEMENTED$'
        DEFB 'NOT IMPLEMENTED$NOT IMPLEMENTED$'
        DEFB 'NOT IMPLEMENTED$NOT IMPLEMENTED$'
        DEFB 'NOT IMPLEMENTED$NOT IMPLEMENTED$'
        DEFB 'NOT IMPLEMENTED$NOT IMPLEMENTED$'
        DEFB 'MENU 1$'
EDIT:   LD IX,HARRY1        ;IX È IL PUNTATORE AL VETTORE
        LD A,(SPX)
        DEC A
        LD C,A
        LD B ,0
        ADD IX,BC           ;SOMMA L'OFFSET DI X
        LD A,(SPY)          ;PRENDE L'OFFSET DI Y
        SLA A
        SLA A
        SLA A               ;8 VOLTE
        LD C,A
        ADD IX,BC           ;SOMMA L'OFFSET DI Y
        LD A,(IX+0)
        INC A
        AND 1
        LD (IX+0),A         ;INVERTE ALLOCAZIONE
                            ;DEL VETTORE
        RET
DOFUNC: LD A,0
        CALL GTTRIG         ;IL TASTO SPAZIATURA
                            ;È PREMUTO?
        CP 255
        RET NZ              ;SE NO RETURN
        LD A,(SPX)
        CP 0
        JR Z,DOFN1
        CP 9
        JR NZ,EDIT          ;SE NON È IN UNA COLONNA
                            ;DI FUNZIONE
DOFN1:  LD IX,DFUNC
```

```
          LD A,(MENU)
          SLA A
          SLA A
          SLA A
          SLA A
          ADD A,(IX+0)
          SLA A
          LD C,A
          LD B,0
          LD IX,FNTABL
          ADD IX,BC
          LD A,(IX+0)             ;PRENDE L'INDIRIZZO
          LD L,A                  ;DALLA TABELLA DEI SALTI
          LD A,(IX+1)             ;IN
          LD H,A                  ;HL
          JP (HL)                 ;E VA AD ESEGUIRE
FNTABL:   DEFW EXIT,LOAD,SAVE,CLR,CHAR,SPRIT
                                  ;TABELLA DEI SALTI A FUNZIONE
          DEFW COLR,LSCRL,RSCRL,USCRL
          DEFW DSCRL,COPY,INVT,FVER,FHOR,NXTMNU
          DEFW POSIT,BANKSW,MSP
          DEFW DMSP,SP16,SP8,NIMP,NIMP,NIMP,NIMP
          DEFW NIMP,NIMP,NIMP,NIMP,NIMP,NXTMNU
NXTMNU:   LD A,(MENU)             ;INVERTE IL MENÙ
          INC A
          AND 1
          LD (MENU),A
NIMP:     RET                     ;ROUTINE FITTIZIA
SPRIT:    LD HL,SPTTAB            ;ROUTINE DI SCELTA
                                  ;DELLO SPRITE
          LD (CHSP)               ;DA DEFINIRE
          JR CHAR1                ;IMPOSTANDO L'INDIRIZZO
                                  ;DI SPRITE
CHARF:    DEFB 0                  ;E SELEZIONANDO
                                  ;IL CARATTERE
CHAR:     LD HL,PATTAB            ;ROUTINE DI SELEZIONE
                                  ;DEL CARATTERE
          LD (CHSP),HL            ;INDIRIZZO DI TABELLA
                                  ;DEL CARATTERE O SPRITE
CHAR1:    CALL GETNUM             ;ROUTINE DI INPUT NUMERICO
          LD (CHARF),A            ;MEMORIZZA IL NUMERO
                                  ;DI CARATTERE
```

```
         CALL TCHAR          ;LO PRENDE PER DEFINIRE
                             ;LA MATRICE
YOBBO:   LD A,0
         CALL GTTRIG
         CP 0
         JR NZ,YOBBO
         RET
GETNUM:  CALL SPPNT          ;STAMPA SPAZI
GN1:     LD HL,NAMTAB+519
         CALL NUMLOP
         INC HL
         LD B,A
         CALL NUMLOP
         SLA B
         SLA B
         SLA B
         SLA B
         ADD A,B             ;PRENDE DUE CARATTERI
                             ;NUMERICI ESADECIMALI
         RET                 ;IN A E RETURN
NUMLOP:  PUSH HL
NUMLAP:  LD C,0
         LD A,0
         CALL GTTRIG
         CP 0
         JR NZ,NUMLAP        ;ATTESA DI RILASCIO DI SPAZIO
NUMLLP:  POP HL
SGLOP:   CALL NPNT           ;STAMPA CIFRA
         PUSH HL
         LD A,0
         CALL GTSTCK
         CP 0
         JR Z,NUMELP
         INC C
         LD A,C
         CP 16
         JR NZ,NUMELP
         LD C,0              ;SE IL TASTO DI UN CURSORE
                             ;È SCHIACCIATO
NUMELP:  HALT
         HALT
         HALT
```

```
        HALT                ;ASPETTA UN BIT
        LD A,0
        CALL GTTRIG         ;FINITO PER CIFRA CORRENTE?
        CP 255
        JR NZ,UMLLP         ;SE NO FALLO ANCORA
        POP HL              ;ALTRIMENTI RIPRISTINA HL
        LD A,C              ;DIGITALE IN A
        RET                 ;RETURN
NPNT:   LD A,C              ;NUMERO IN A
        ADD A,48            ;CONVERTE IN CODICE ASCII
        CP 58               ;SE ESADECIMALE DA A AD F
        JR C,NPNT2
        ADD A,7             ;ALLORA CONVERTE
                            ;ULTERIORMENTE IN ASCII
NPNT2:  CALL WRTVRM         ;LO STAMPA
        RET
TCHAR:  LD A,(CHARF)        ;QUESTA ROUTINE
                            ;TRASFERISCE
        LD L,A              ;LA DEFINIZIONE IN VRAM
                            ;DEL CARATTERE OPPURE
        LD H,0              ;DELLO SPRITE NELLA MATRICE
                            ;DI DEFINIZIONE
        ADD HL,HL
        ADD HL,HL
        ADD HL,HL
        PUSH HL
        POP DE
        LD IX,(CHSP)
        ADD IX,DE
        PUSH IX
        POP HL
        LD C,8
        LD IY,HARRY1
TCOLP:  LD B,8
        CALL RDVRM
TCILP:  LD D,0
        RLC A               ;SPOSTA I BIT DI DEFINIZIONE
        JR NC,TCSKP         ;NEL CARRY
        LD D,1              ;E SCRIVE 0 E 1
TCSKP:  LD (IY+0),D         ;NEL VETTORE DOVE
                            ;NECESSARIO
        INC IY              ;INCREMENTA IL PUNTATORE
```

```
                                        ;AL VETTORE
         DJNZ TCILP                     ;E RIPETE
         INC HL                         ;INCREMENTA IL PUNTATORE
                                        ;ALLA VRAM
         DEC C
         JR NZ,TCOLP                    ;E LO FA PER 8 RIGHE
         RET
CHSP:    DEFW PATTAB
CHART:   LD A,(CHARF)                   ;QUESTA ROUTINE È
         LD L,A                         ;L'INVERSO
         LD H,0                         ;DI QUELLA PRECEDENTE
         ADD HL,HL
         ADD HL,HL
         ADD HL,HL
         PUSH HL
         POP DE
         LD IX,(CHSP)
         ADD IX,DE
         PUSH IX
         POP HL
         LD C,8
         LD IY,HARRY1
CTLO:    XOR A
         LD D,A
         LD B,8
CTLI:    LD A,(IY+0)
         CP1
         CCF
         RL D
         INC IY
         DJNZ CTLI
         LD A,D
         CALL WRTVRM
         INC HL
         DEC C
         JR NZ,CTLO
         RET
CLR:     LD IX,HARRY1                   ;ROUTINE CHE CANCELLA
                                        ;LA MATRICE
         LD B,64                        ;VETTORE DI 64 BYTE
         LD A,0                         ;0 CANCELLA
CLRLP:   LD (IX+0),A                    ;LA MATRICE
```

```
        INC IX          ;INCREMENTA IL PUNTATORE
                        ;ALLA MATRICE
        DJNZ CLRLP      ;64 VOLTE
        RET             ;RETURN
INVT:   LD IX,HARRY1    ;IX È IL PUNTATORE
                        ;ALLA MATRICE
        LD B,64         ;LA MATRICE È LUNGA 64 BYTE
IVTLP:  LD A,(IX+0)     ;PRENDE I CONTENUTI
                        ;DELLA MATRICE
        INC A           ;0 DIVENTA 1
        AND 1           ;E 1 DIVENTA 0
        LD (IX+0),A     ;LO RIMETTE A POSTO
        INC IX          ;INCREMENTA IL PUNTATORE
        DJNZ IVTLP      ;LO FA 64 VOLTE
        RET
POSIT:  CALL CURPOS     ;POSIZIONE DEL CURSORE
        LD HL,(CHSP)    ;CARATTERE O SPRITE?
        XOR A           ;CANCELLA A E IL FLAG
                        ;DI RIPORTO
        LD DE,PATTAB    ;INDIRIZZO DELLA TABELLA
                        ;DI GEN. DEI MODELLI
        SBC HL,DE       ;STIAMO FACENDO
                        ;UN CARATTERE?
        JR NZ,SPOSIT    ;NO, VAI A POSIZIONARE
                        ;LO SPRITE

        LD A,(CYP)
        AND 0F8H
        SLA A
        SLA A           ;DIVIDE PER 8 X E Y
                        ;DEL CURSORE
        LD B,A
        LD A,(CXP)
        SRL A
        SRL A
        SRL A
        ADD A,B
        LD HL,NAMTAB
        LD E,A
        LD D,0
        ADD HL,DE       ;CALCOLA L'INDIRIZZO
                        ;DELLA TABELLA DEI NOMI
        LD A,(CHARF)
```

```
        CALL WRTVRM         ;E SCRIVE IL CARATTERE
                            ;CORRENTE
        RET
SPOSIT: LD HL,SATTAB        ;ROUTINE CHE POSIZIONA
                            ;SPRITE
        LD A,(CHARF)        ;NUMERO DI SPRITE
        AND 31              ;MOD 32
        LD B,A
SPOST0: INC HL
        INC HL
        INC HL
        INC HL
        DJNZ SPOST0         ;PIANO DI SPRITE DA USARE
SPOST1: LD A,(CYP)
        CALL WRTVRM
        INC HL
        LD A,(CXP)
        CALL WRTVRM
        INC HL
        LD A,(CHARF)
        CALL WRTVRM
        LD A,(SPCOL)
        INC HL
        CALL WRTVRM         ;SCRIVE GLI ATTRIBUTI
                            ;DI SPRITE
        RET
SPCOL:  DEFB 3              ;MEMORIZZA I COLORI
                            ;DI SPRITE
CHCOL:  DEFB 31H,31H,31H,31H,31H,31H,31H,31H
                            ;COLORI CARATTERISTICI
CXP:    DEFB 0
CYP:    DEFB 0
CURPOS: LD A,16             ;POSIZIONE DI PARTENZA PER...
        LD (CXP),A          ;POSIZIONARE IL CURSORE
        LD (CYP),A
CUP1:   LD A,0
        CALL GTTRIG         ;ABBIAMO LIBERATO
        CP 255              ;IL TASTO SPAZIATURA
        JR Z,CUP1
CUPOLP: LD HL,(CXP)         ;PRENDE LE COORDINATE X, Y
                            ;DEL CURSORE
        LD A,0
```

```
        PUSH HL
        CALL GTSTCK
        POP HL
        CP 1
        JR NZ,CUP2          ;MUOVE IL CURSORE
        DEC H
CUP2:   CP 3
        JR NZ,CUP3
        INC L
CUP3:   CP 5
        JR NZ,CUP4
        INC H
CUP4:   CP 7
        JR NZ,CUP5
        DEC L
CUP5:   LD A,H
        CP 57
        JR C,CUP6
        LD H,0
CUP6:   LD (CXP),HL
        LD HL,SATTAB
        LD A,(CYP)
        CALL WRTVRM
        INC HL
        LD A,(CXP)
        CALL WRTVRM
        INC HL
        INC HL
        LD A,12
        CALL WRTVRM         ;SCRIVE LA NUOVA POSIZIONE
                            ;DI SPRITE
        LD A,0
        PUSH HL
        CALL GTTRIG         ;ABBIAMO FINITO I MOVIMENTI?
        POP HL
        HALT
        HALT
        HALT
        HALT
        CP 0
        JR Z,CUPOLP         ;SE NO MUOVERE ANCORA
        RET
```

```
CSPT:      LD B,20
           LD HL,NAMTAB+547
CSPT0:     LD A,32
           CALL WRTVRM
           INC HL
           DJNZ CSPT0
           LD HL,(CHSP)
           LD DE,PATTAB          ;QUESTA ROUTINE STAMPA...
           LD IX,CMESS           ;TUTTO LO SPAZIO RIMANENTE...
           SBC HL,DE             ;ED IL MESSAGGIO
           JR Z,CSPT1
           LD IX,SMESS
CSPT1:     LD HL,NAMTAB+547
CSPT2:     LD A,(IX+0)
           CP '$'
           JR Z,CSPT3
           CALL WRTVRM
           INC HL
           INC IX
           JR CSPT2
CSPT3:     LD A,(CHARF)
           SRL A
           SRL A
           SRL A
           SRL A
           LD C,A
           CALL NPNT
           INC HL
           LD A,(CHARF)
           AND 15
           LD C,A
           CALL NPNT             ;QUESTE ROUTINE
                                 ;MANTENGONO...
           RET                   ;AGGIORNATA L'AREA STATUS...
CMESS:     DEFB 'CHAR:$'         ;DELLO SCHERMO
SMESS:     DEFB 'SPRITE:$'
COLMES:    DEFB 'SPRITECOL:$'
COLMS1:    DEFB 'CHARCOLS$'
COLPT:     LD HL,NAMTAB+579
           LD IX,COLMES
COLPT1:    LD A,(IX+0)
           CP '$'
```

```
        JR Z,COLPT2
        CALL WRTVRM
        INC HL
        INC IX
        JR COLPT1
COLPT2: LD A,(SPCOL)
        SRL A
        SRL A
        SRL A
        SRL A
        LD (SPCOL),A
        CALL NPNT
        LD A,(SPCOL)
        AND 15
        LD C,A
        INC HL
        CALL NPNT
        LD HL,NAMTAB+532
        LD IX,COLMS1
COLPT3: LD A,(IX+0)
        CP '$'
        JR Z,COLPT4
        CALL WRTVRM
        INC HL
        INC IX
        JR COLPT3
COLPT4: INC HL
        PUSH HL
        LD A,(CHARF)
        LD L,A
        LD H,0
        ADD HL,HL
        ADD HL,HL
        ADD HL,HL
        PUSH HL
        POP DE
        LD IX,COLTAB
        ADD IX,DE
        PUSH IX
        POP HL
        LD IX,CHCOL
        LD B,8
```

```
COLPT5:     CALL RDVRM
            LD (IX+0),A
            INC IX
            INC HL
            DJNZ COLPT5
            POP HL
            LD B,8
            LD DE,31
            LD IX,CHCOL
COLPT6:     LD A,(IX+0)
            PUSH AF
            SRL A
            SRL A
            SRL A
            SRL A
            LD C,A
            CALL NPNT
            POP AF
            AND 15
            INC HL
            LD C,A
            CALL NPNT
            ADD HL,DE
            INC IX
            DJNZ COLPT6
            RET
BANK:       DEFB 0
BPMESS:     DEFB 'BANK'
BANKPT:     LD HL,NAMTAB+611
            LD IX,BPMESS
BKPT1:      LD A,(IX+0)
            CP '$'
            JR Z,BKPT2
            CALL WRTVRM
            INC HL
            INC IX
            JR BKPT1
BKPT2:      LD A,(BANK)
            AND 15
            LD C,A
            CALL NPNT
            RET
```

```
FTTXRM:   LD HL,PATTAB
          LD DE,0B000H
          LD BC,1800H
          CALL LDIRMV
          LD HL,COLTAB
          LD DE,0C800H
          LD BC,1800H
          CALL LDIRMV
SPTXRM:   LD HL,SPTTAB
          LD DE,0E000H
          LD BC,800H
          CALL LDIRMV
          RET
LPTXRM:   LD A,(BANK)
          LD H,A
          LD L,0
          ADD HL,HL
          ADD HL,HL
          ADD HL,HL
          PUSH HL
          LD DE,0B000H
          ADD HL,DE
          EX DE,HL
          LD HL,PATTAB
          LD BC,800H
          CALL LDIRMV
          POP HL
          LD DE,0C800H
          ADD HL,DE
          EX DE,HL
          LD HL,COLTAB
          LD BC,800H
          CALL LDIRMV
          JR SPTXRM
COLR:     CALL SPPNT
          LD A,0
          CALL GTTRIG
          CP 255
          JR Z,COLR
          LD HL,(CHSP)
          XOR A
          LD DE,PATTAB
```

```
        SBC HL,DE
        JR NZ,COLSP
        LD IX,CHCOL
        LD B,8
COLLYP: LD A,8
        SUB B
        ADD A,48
        PUSH BC
        LD HL,NAMTAB+517
        CALL WRTVRM
        INC HL
        LD A,':'
        CALL WRTVRM
        INC HL
        LD A,(IX+0)
        SRL A
        SRL A
        SRL A
        SRL A
        LD C,A
HNGON1: LD A,0
        CALL GTTRIG
        CP 255
        JR Z,HNGON1
        PUSH IX
        CALL SGLOP
        POP IX
        SLA A
        SLA A
        SLA A
        SLA A
        LD B,A
        LD A,(IX+0)
        AND 15
        OR B
        LD (IX+0),A
        AND 15
        LD C,A
        INC HL
HANGON: LD A,0
        CALL GTTRIG
        CP 255
```

```
        JR Z,HANGON
        PUSH IX
        CALL SGLOP
        POP IX
        AND 15
        LD B,A
        LD A,(IX+0)
        AND 0F0H
        OR B
        LD (IX+0),A
        INC IX
        POP BC
        DJNZ COLLYP
        LD A,(CHARF)
        LD L,A
        LD H,0
        ADD HL,HL
        ADD HL,HL
        ADD HL,HL
        LD DE,COLTAB
        ADD HL,DE
        EX DE,HL
        LD HL,CHCOL
        LD BC,8
        CALL LDIRVM
        RET
COLSP:  CALL GETNUM
        LD (SPCOL),A
        RET
DMSP:   LD A,(9001H)        ;PRENDE IL VALORE ATTUALE
                            ;DI VDP(1)
        OR 1                ;PONE A 1 IL FLAG MAG
        LD (9001H),A        ;LO RIMETTE A POSTO
        LD B,A
        LD C,1
        CALL WRTVRM         ;LO SCRIVE NEL VDP
        RET
MSP:    LD A,(9001H)
        AND 0FEH            ;RIMETTE A POSTO IL FLAG MAG
        LD (9001H),A
        LD B,A
        LD C,1
```

```
        CALL WRTVDP        ;LO SCRIVE IN VDP
        RET
SP8:    LD A,(9001H)
        AND 0FDH           ;PONE A 0 FLAG DI SPRITE
                           ;16 * 16
        LD (9001H),A
        LD B,A
        LD C,1
        CALL WRTVDP        ;E LO SCRIVE NEL VDP
        RET
SP16:   LD A,(9001H)
        OR 2               ;PONE A 1 FLAG DI SPRITE
                           ;16 * 16
        LD (9001H),A
        LD B,A
        LD C,1
        CALL WRTVDP        ;E LO SCRIVE NEL VDP
        RET
LSCRL:  LD IX,HARRY1       ;SPOSTA A SINISTRA
                           ;DAL VETTORE 1
        LD IY,HARRY2       ;AL VETTORE 2
        LD C,8
LSLP:   LD A,(IX+0)
        LD (IY+7),A
        LD B,7
        INC IX
LSLP1:  LD A,(IX+0)
        LD (IY+0),A
        INC IX
        INC IY
        DJNZ LSLP1
        INC IY
        DEC C
        JR NZ,LSLP
        LD IX,HARRY1       ;METTE IL VETTORE 2
                           ;NEL VETTORE 1
        LD IY,HARRY2
        LD B,64
LSLP2:  LD A,(IY+0)
        LD (IX+0),A
        INC IX
        INC IY
```

```
        DJNZ LSLP2          ;64 BYTE?
LSLP3:  LD A,0
        CALL GTTRIG         ;ASPETTA CHE SI LIBERI
                            ;IL TASTO SPAZIATURA
        CP 255
        JR Z,LSLP3
        RET
RSCRL:  LD B,7
RSCRLP: PUSH BC
        CALL LSCRL
        POP BC
        DJNZ RSCRLP
        RET
USCRL:  LD DE,8             ;SPOSTAMENTO IN ALTO
                            ;È SIMILE...
        LD IX,HARRY1        ;A SPOSTAMENTO SINISTRO
        LD IY,HARRY2
        LD C,8
USRLP1: PUSH IX
        PUSH IY
        LD A,(IX+0)
        LD (IY+56),A
        ADD IX,DE
        LD B,7
USRLP2: LD A,(IX+0)
        LD (IY+0),A
        ADD IX,DE
        ADD IY,DE
        DJNZ USRLP2
        POP IY
        POP IX
        INC IY
        INC IX
        DEC C
        JR NZ,USRLP1
        LD IX,HARRY1
        LD IY,HARRY2
        LD B,64
USRLP3: LD A,(IY+0)
        LD (IX+0),A
        INC IX
        INC IY
```

```
        DJNZ USRLP3
USRLP4: LD A,0
        CALL GTTRIG
        CP 255
        JR Z,USRLP4
        RET
DSCRL:  LD B,7              ;SPOSTAMENTO VERSO
                            ;IL BASSO
DCRLP:  PUSH BC
        CALL USCRL          ;7 VOLTE
        POP BC
        DJNZ DCRLP
        RET
FVER:   LD IX,HARRY1        ;RIFLESSIONE VERTICALE
                            ;DAL VETTORE 1
        LD IY,HARRY2+56     ;AL VETTORE 2
        LD C,8
        LD DE,8
FVER1:  PUSH IX
        PUSH IY
        LD B,8
FVER2:  LD A,(IX+0)
        LD (IY+0),A
        INC IX
        INC IY
        DJNZ FVER2
        POP HL
        XOR A
        SBC HL,DE
        PUSH HL
        POP IY
        POP IX
        ADD IX,DE
        DEC C
        JR NZ,FVER1
        LD B,64
        LD IX,HARRY1        ;RIPORTA POI IL VETTORE 2
                            ;MODIFICATO...
        LD IY,HARRY2        ;NEL VETTORE 1
FVER3:  LD A,(IY+0)
        LD (IX+0),A
        INC IX
```

```
        INC IY
        DJNZ FVER3
        JP USRLP4
FHOR:   LD IX,HARRY1        ;LA RIFLESSIONE
                            ;ORIZZONTALE...
        LD IY,HARRY2+7      ;È SIMILE A QUELLA VERTICALE
        LD C,8
FHOR1:  PUSH IY
        LD DE,8
        LD B,8
FHOR2:  LD A,(IX+0)
        LD (IY+0),A
        INC IX
        DEC IY
        DJNZ FHOR2
        POP IY
        XOR A
        ADD IY,DE
        DEC C
        JR NZ,FHOR1
        LD B,64
        LD IX,HARRY1
        LD IY,HARRY2
FHOR3:  LD A,(IY+0)
        LD (IX+0),A
        INC IX
        INC IY
        DJNZ FHOR3
        JP USRL4
BANKSW: CALL SPPNT
        LD HL,NAMTAB+157
        CALL NUMLOP
        CP 3
        JR NC,BANKSW
BSW1:   LD (BANK),A
        LD BC,800H
        LD DE,0B000H
        LD H,A
        LD L,0
        ADD HL,HL
        ADD HL,HL
        ADD HL,HL
```

```
         PUSH HL
         LD HL,DE
         LD DE,PATTAB
         CALL LDIRVM
         POP HL
         LD DE,0C800H
         ADD HL,DE
         LD DE,COLTAB
         LD BC,800H
         CALL LDIRVM
         CALL TCHAR
         RET
COPBNK:  DEFB 0
COPCHA:  DEFB 0
COPY:    CALL SPPNT          ;ROUTINE CHE COPIA
                             ;UNA DEFINIZIONE
         LD IX,BPMESS
         LD HL,NAMTAB+517
COPYLP:  LD A,(IX+0)
         CP '$'
         JR Z,COPY1
         CALL WRTVRM
         INC HL
         INC IX
         JR COPYLP           ;STAMPA MESSAGGI
COPY1:   CALL NUMLOP         ;PRENDE IL NUMERO DI BANCO
         LD (COPBNK),A       ;SALVA IL BANCO DA COPIARE
         CALL GETNUM         ;PRENDE IL NUMERO
                             ;DI CARATTERE/SPRITE
         LD (COPCHA),A
         LD A,(COPBNK)
         CP 4
         JR NC,COPY
         CP 3
         JP Z,SPCOPY         ;COPIA CARATTERI O SPRITE
         LD A,(COPCHA)
         LD L,A
         LD A,(COPBNK)
         LD H,A
         ADD HL,HL
         ADD HL,HL
```

```
        ADD HL,HL           ;CALCOLA L'INDIRIZZO
                            ;DEL CARATTERE
        PUSH HL
        LD DE,0B000H
        ADD HL,DE           ;NELLA RAM
        PUSH HL
        LD A,(CHARF)
        LD L,A
        LD H,0
        ADD HL,HL
        ADD HL,HL
        ADD HL,HL
        LD DE,(CHSP)
        ADD HL,DE
        EX DE,HL            ;PRENDE L'INDIRIZZO IN VRAM
                            ;DEL CARATTERE O DELLO SPRITE
        POP HL
        LD BC,8
        CALL LDIRVM         ;LO TRASFERISCE DALLA RAM
                            ;ALLA VRAM
        POP BC
        LD DE,(CHSP)
        LD HL,PATTAB
        XOR A
        SBC HL,DE
        JR NZ,COPSKP
        PUSH BC
        POP HL
        LD DE,0C800H        ;SE STA TRASFERENDO
                            ;UN CARATTERE
        ADD HL,DE
        PUSH HL
        LD DE,COLTAB
        LD A,(CHARF)
        LD L,A
        LD H,0
        ADD HL,HL
        ADD HL,HL
        ADD HL,HL
        ADD HL,HL
        EX DE,HL
        POP HL
```

```
            LD BC,8
            CALL LDIRVM         ;PRENDE ELEMENTO
                                ;DELLA TABELLA DEI COLORI
COPSKP:     CALL TCHAR          ;METTE IL CARATTERE
                                ;NELLA MATRICE
            RET
SPCOPY:     LD A,(COPCHA)       ;SUBROUTINE CHE COPIA
                                ;SPRITE
            LD L,A
            LD H,0
            ADD HL,HL
            ADD HL,HL
            ADD HL,HL
            LD DE,0E000H
            ADD HL,DE
            PUSH HL
            LD A,(CHARF)
            LD L,A
            LD H,0
            ADD HL,HL
            ADD HL,HL
            ADD HL,HL
            LD DE,(CHSP)
            ADD HL,DE
            EX DE,HL
            POP HL
            LD BC,8
            CALL LDIRVM
            CALL TCHAR
            RET
LOAD:       LD A,1              ;LOAD È LA FUNZIONE BASIC 1
L1:         LD (BASFNC),A
            POP HL              ;PRENDE L'INDIRIZZO
                                ;DI RITORNO DALLO STACK
            LD HL,L2            ;PRENDE IL NUOVO INDIRIZZO
                                ;DI RIENTRO
            LD (RTADDR),HL      ;LO SALVA PER IL BASIC
            RET                 ;AL BASIC
L2:         LD A,(BANK)
            CALL GTX
            CALL TCHAR
            JP NDFNC
```

```
SAVE:     LD A,2            ;SAVE È LA FUNZIONE BASIC 2
          JR L1             ;VA ALLA ROUTINE LOAD
GTX:      LD HL,0B000H
          LD DE,PATTAB
          LD BC,1800H
          CALL LDIRVM
          LD HL,0C800H
          LD DE,COLTAB
          LD BC,1800H
          CALL LDIRVM
          LD HL,0E000H
          LD DE,SPTTAB
          LD BC,800H
          CALL LDIRVM
          RET
BASFNC:   DEFB 0            ;MEMORIZZA IL NUMERO
                            ;DI FUNZIONE
RTADDR:   DEFW 0            ;E L'INDIRIZZO DI RITORNO
                            ;PER IL BASIC
          END
```

Il programma riportato deve essere assemblato e salvato su nastro con nome 'GM2'. Si digiti ora il programma BASIC che segue e lo si salvi a sua volta su nastro. Questo programma ha la funzione di caricare e far eseguire il codice macchina.

```
 10 CLEAR 200,&H8FFFF
 20 PRINT "LOADING..."
 30 BLOAD"CAS:GM2"
 40 DEFUSR=&H9008
 50 A=USR(A)
 60 DEFUSR=PEEK(&H9AED)+256*(PEEK(&H9AEE)):REM RTADDR
 70 IF PEEK(&H9AEC)=0 THEN SCREEN1,0,0,1:END
 80 IF PEEK(&H9AEC)=1 THEN BLOAD"CAS:CHARS"
 90 IF PEEK(&H9AEC)=2 THEN BSAVE"CAS:CHARS",&HB000,&HE801
100 GOTO 50
```

**Uso del programma definitore**

Quando manderete in esecuzione il programma vi accorgerete che lo schermo è diviso in tre zone. La zona più alta è nera, la zona intermedia contiene una griglia di 10 * 8 elementi (le cui colonne estreme di destra e sinistra sono segnate con una linea di piccole croci) e nella terza zona vengono vi-

sualizzate le funzioni che il programma è in grado di svolgere. Muovendo il cursore sulle colonne della griglia contenute nella seconda zona vedrete apparire dei messaggi sotto la griglia, nella terza sezione dello schermo. Il movimento del cursore avviene per mezzo dei tasti di controllo del cursore stesso, mentre la barra di spaziatura ha una doppia funzione: quando il cursore si trova nella parte centrale (8 * 8) della matrice di definizione, la pressione della barra di spaziatura comporta il cambiamento di stato della posizione in cui si trova il cursore. Quando il cursore è invece in una delle colonne riservate alle funzioni, la pressione della barra di spaziatura provoca l'attivazione della funzione stessa, scelta tra quelle visualizzate sotto la matrice di definizione.
Oltre ai messaggi delle funzioni, la terza parte dello schermo, quella inferiore, viene utilizzata anche per alcuni messaggi di stato, la maggior parte dei quali sono autoesplicativi.
Le funzioni che il programma può svolgere sono le seguenti:

| | |
|---|---|
| EXIT | Termina il programma. |
| LOAD | Carica da nastro un file chiamato "CHARS" che contiene la definizione dei caratteri, quella degli sprite e i colori dei caratteri. |
| SAVE | Salva il file "CHARS" su nastro. |
| CLEAR | Azzera la matrice di definizione. |
| CHARACTER | Seleziona il carattere che deve essere modificato e trasferisce la sua definizione dalla VRAM alla matrice di definizione. |
| SPRITE | Seleziona lo sprite che deve essere definito. |
| COLOUR | In fase di definizione di un carattere, questa funzione consente di definire il suo colore riga per riga. In fase di definizione di uno sprite imposta il colore per la funzione POSITION. |
| SCROLL LEFT<br>RIGHT<br>UP<br>DOWN | Sposta la definizione corrente sulla matrice nella direzione indicata; le parti che scompiaiono sulla destra ricompaiono sulla sinistra e viceversa (analogamente per l'alto e il basso). |
| COPY | Copia una definizione di carattere o di sprite in quella corrente. |
| INVERT | Cambia lo stato di tutti i pixel della definizione corrente. |
| FLIP<br>HORIZONTAL<br>VERTICAL | Produce una definizione speculare di quella corrente rispetto a una linea (orizzontale o verticale) che passa per il centro della matrice. |
| MENU 1/2 | Scambia i menù di funzioni accessibili partendo dalle colonne riservate alle funzioni. |

| | |
|---|---|
| POSITION | Consente di posizionare caratteri e sprite nella parte superiore dello schermo. |
| SWAP BANK | Come abbiamo visto in precedenza, la tabella dei modelli nella modalità Grafica II consiste di tre insiemi di 256 definizioni ciascuna. Il numero di banco determina a quale di questi tre insiemi appartengono i caratteri che si stanno definendo e quelli che compaiono sullo schermo (si noti che quando viene chiesto il numero di banco nella funzione COPY, i numeri da 0 a 2 indicano i caratteri, il 3 la definizione degli sprite). |
| MAGNIFY<br>DEMAGNIFY<br>SPRITES | Queste due funzioni impostano i registri del VDP determinando il fattore di ingrandimento degli sprite. |
| 8*8/16*16 SPRITES | |

Un input numerico viene sempre considerato sotto forma di numero esadecimale di due cifre. Le cifre vengono selezionate dal cursore; quella più significativa precede quella di ordine inferiore.
Le definizioni dei caratteri, al termine del programma, vengono lasciate nella RAM della CPU nelle seguenti locazioni:

| | |
|---|---|
| Da 0B000H a 0C800H: | la tabella di generazione dei modelli (la definizione dei caratteri). |
| Da 0C800H a 0E000H: | la tabella dei colori. |
| Da 0E000H a 0E800H: | le definizioni degli sprite. |

**Il definitore nella altre modalità video**
Nella modalità Grafica I vi sono solo 256 possibili definizioni di caratteri e quindi è necessaria la definizione solo del primo "banco". Esso risiederà quindi dalla locazione 0B000H alla 0B800H (la tabella dei colori può essere trascurata); le definizioni degli sprite restano attive.
Se si usa il definitore nella modalità testo rimangono valide le osservazioni fatte, con l'eccezione che, non essendo possibile il loro utilizzo, le definizioni degli sprite risultano irrilevanti. Va ricordato che nella modalità testo vengono usate solo le 6 colonne più a sinistra: è quindi consigliabile lasciare in bianco le altre due colonne di ogni definizione.

## Definizione dinamica di modelli

Poiché il VDP tiene nella VRAM tutte le definizioni dei caratteri, è possi-

bile ottenere una sorta di animazione nel piano dei modelli ridefinendo i caratteri durante l'esecuzione del programma. Quando viene fatto ciò, si può notare che la nuova definizione viene applicata a tutte le occorrenze del carattere che si è modificato. In questo modo è possibile modificare ampie zone dello schermo variando un numero relativamente piccolo di elementi della tabella di generazione dei modelli. Consideriamo, a titolo esemplificativo, la definizione del disegno di una scala a pioli largo 7 caratteri e alto 1. Esso apparirà simile ad un elevatore con gli elementi mobili che si spostano dal basso verso l'alto: questo effetto può essere ottenuto agendo su un solo carattere.
Prima di tutto definiamo il carattere di base. Per ottenere il disegno che ci prefiggiamo, abbiamo bisogno di un carattere che sia come un pezzo della scala; quest'ultima verrà poi ottenuta "impilando" più caratteri come quello che stiamo definendo. Ad esempio può andar bene il seguente modello:

```
10000001  81H
11111111  FFH
10000001  81H
10000001  81H
10000001  81H
11111111  FFH
10000001  81H
10000001  81H
```

Dopo aver deciso il modello del carattere, è necessario scriverlo nella VRAM: lo facciamo indicandolo come carattere 0.

```
CHRDEF:   DEFB 081H,0FFH,081H,081H,081H,0FFH,081H,081H
CHARIN:   LD B,8              ;NUMERO DEGLI ELEMENTI
                              ;IN CHRDEF
          LD IX,CHRDEF        ;PORTA L'INDIRIZZO
                              ;DELLA DEFINIZIONE IN IX
          LD HL,PGTAB         ;PORTA IN HL L'INDIRIZZO
                              ;DELLA TABELLA DI GEN. MOD.
CILP:     LD A,(IX+0)         ;PRENDE UN BYTE DATO
          CALL 004DH          ;SCRIVE (HL) NELLA VRAM
          INC HL
          INC IX              ;INCREMENTA I PUNTATORI
          DJNZ CILP           ;LO FA 8 VOLTE
          RET
```

Abbiamo ora nella VRAM una definizione che possiamo variare per otte-

nere l'effetto desiderato. La manipolazione richiesta è uno spostamento verticale degli otto byte della definizione del carattere, facendo ricomparire in basso la riga che esce, "sospinta" dalle altre, dall'alto. La routine che segue ottiene quanto detto:

```
        RDVRM  EQU 004AH
        WRTVRM EQU 004DH
DSCROL: LD B,7          ;IMPOSTA IL CONTATORE
                        ;DEL CICLO
        LD HL,PGTAB     ;IMPOSTA L'IND. DI VRAM
                        ;DELLA DEFINIZIONE
        CALL RDVRM      ;LEGGE IL PRIMO VALORE
        LD C,A          ;LO SALVA PER DOPO
DSCLP:  INC HL          ;INCREMENTA IL PUNTATORE
                        ;IN VRAM
        CALL RDVRM      ;LEGGE UNA LINEA
                        ;DELLA DEFINIZIONE
        DEC HL          ;E LA MEMORIZZA UNA
        CALL WRTVRM     ;POSIZIONE PIÙ IN ALTO
        INC HL          ;RIMPOSTA IL PUNTATORE
                        ;IN VRAM
        DJNZ DSCLP      ;LO FA 7 VOLTE
        LD A,C          ;LA VECCHIA PRIMA LINEA
        CALL WRTVRM     ;DIVENTA QUELLA DI FONDO
        RET
```

Chiamando questa procedura più volte la nostra scala sembrerà avere un movimento verticale, dal basso verso l'alto, dei pioli. Possono essere costruiti disegni di scale di qualunque dimensione; è necessario anche il seguente programma:

```
START:  CALL CHARIN     ;DEFINISCE IL CARATTERE
                        ;CHIAMANDO LA ROUTINE
                        ;PRECEDENTE
        LD HL,NAMTAB+8  ;PRENDE L'INDIRIZZO
                        ;DELLA TAB. NOMI + 8
        LD DE,32        ;LUNGHEZZA DI LINEA IN DE
        LD B,7          ;LA SCALA È LUNGA
                        ;7 CARATTERI
PLOOP:  CALL WRTVRM     ;STAMPA IL CARATTERE SCALA
        ADD HL,DE       ;DECREMENTA DI 1
                        ;LA POSIZIONE DI STAMPA
        DJNZ PLOOP      ;LO FA PER 7 CARATTERI
```

```
LOOP2:      CALL DSCROL      ;SPOSTA LA DEFINIZIONE
            HALT
            HALT             ;PAUSA DI 2/50 DI SECONDO
            JR LOOP2         ;LO FA NUOVAMENTE
```

Come abbiamo potuto capire da questo esempio, il principio della definizione dinamica dei caratteri è uno strumento molto potente: il suo utilizzo è limitato solo dall'immaginazione. Infatti è il principio della definizione dinamica dei caratteri che ci consente di considerare (nella prossima sezione) la modalità Grafica II come una modalità "bit mapped".

## La modalità Grafica II come modalità "bit mapped"

Se nella modalità Grafica II impostiamo la tabella dei nomi in modo che i 768 byte della tabella vengano numerati consecutivamente da 0 a 255 per tre volte, ne consegue che ogni singola posizione carattere ha una sua propria ed unica definizione nella tabella di generazione dei modelli, e questa situazione può essere considerata essere "bit mapped", così come mostra la figura 6.1. Avendo preparato la tabella dei nomi nel modo suddetto ora disponiamo, a tutti gli effetti, di una modalità ad alta risoluzione con 256 * 192 pixels e con la possibilità di 15 colori (sebbene disgraziatamente la risoluzione orizzontale del colore è di solo 32 blocchi di otto pixel).
Se consideriamo come origine l'angolo in basso a destra, la formula che permette di calcolare quale byte nella tabella di generazione dei modelli contiene il pixel puntato dalle coordinate X e Y è la seguente:

(Indirizzo inizio della tabella di generazione dei modelli) + ((((191 —Y)div 8)*256)+((191—Y)mod 8)+(X AND 0F8H))

e il bit che deve essere posto a 1 all'interno del byte può essere individuato con:

7—(X AND 7)

Facendo uso delle suddette formule, la procedura che viene riportata di seguito serve per impostare il pixel le cui coordinate sono in HL (L contiene la coordinata X, mentre H la coordinata Y) con il colore specificato dall'accumulatore:

```
SETXY:      PUSH AF          ;SALVA IL CODICE DEL COLORE
            LD A,191         ;PONE Y UGUALE A...
            SUB H            ;191 —Y
            LD H,A           ;LO RIMETTE IN H
```

```
PUSH HL          ;SALVA X,Y
SRL H
SRL H
SRL H            ;Y DIVENTA Y DIV 8
LD D,H
POP HL           ;RECUPERA X,Y
LD A,H           ;PRENDE Y
AND 7            ;MOD 8
LD E,A
XOR A            ;PULISCE IL FLAG DI CARRY
                 ;PER ADDIZIONE 16 BIT
LD A,L           ;PRENDE X
AND 0F8H         ;
LD E,A           ;LO SOMMA A DE
LD A,0
ADC A,D          ;TERMINA ADDIZIONE 16 BIT
LD D,A           ;DE CONTIENE ORA L'OFFSET
                 ;DALLA TAB. GEN. MODELLI,
                 ;E L'INDIRIZZO BASE
                 ;DELLA TABELLA DEI COLORI.
LD B,L           ;SALVA X
LD HL,COLTAB     ;PRENDE LA BASE DELLA TAB.
                 ;COLORI
ADD HL,DE        ;SOMMA L'OFFSET
CALL 004AH       ;PRENDE IL VALORE CORRENTE
                 ;DI COLORE
AND 15           ;MASCHERA LO SFONDO
LD C,A           ;LO SALVA IN C
POP AF           ;RIPRISTINA IL COLORE
SLA A            ;SHIFT DEL CODICE COLORE
SLA A
SLA A
SLA A            ;NEL SEMIBYTE ALTO
ADD A,C          ;SOMMA IL COLORE DELLO
                 ;SFONDO
ALL 004DH        ;E LO RISCRIVE
LD A,7
AND B
LD B,A
XOR A
INC B
CCF
```

```
MLP:    RRA             ;
        DJNZ MLP        ;
        LD HL,PGTAB     ;IND. TAB. GENERAZIONE
                        ;MODELLI IN HL
        ADD HL,DE       ;SOMMA L'OFFSET
        CALL 004AH      ;LEGGE IL VECCHIO VALORE
        XOR B           ;MASCHERA LOGICA CON B
        CALL 004DH      ;LO RISCRIVE
        RET             ;FATTO
```

Considerando le nozioni di base della modalità Grafica II si può intuire che è semplice, manipolando gli elementi nella tabella dei nomi, impostare molte diverse schermate di mappa della memoria. In pratica però la mappa di memoria descritta appena in precedenza e nella figura 6.7 è facile da usare come le altre possibili, ma per molti altri versi è più conveniente. Si deve inoltre ricordare che, sebbene la tecnica che si è ora esaminata ci permetta di considerare la modalità di schermo Grafica II come una modalità "bit mapped", possiamo ancora svolgere le operazioni fondamentali sui caratteri, come se la memoria fosse ancora mappata a caratteri. Quindi l'intero schermo può essere spostato (verso l'alto o il basso) a passi da 8 pixel ciascuno, manipolando semplicemente i 768 byte della VRAM. In alternativa possiamo definire l'ultimo blocco delle definizioni dei caratteri (quelli che corrispondono ai riferimenti della terza parte, la più bassa, della tabella dei nomi) come contenente un insieme di caratteri standard, e trattare i due terzi più alti dello schermo come se fossero "bit mapped". Altre possibilità verranno senz'altro ideate dall'utente.

## Sprite: tecniche d'interruzione

Ci sono due ovvi problemi connessi con la gestione degli sprite da parte del processore video TMS-9929A. Il primo è che ogni singolo sprite deve essere di un solo colore e il secondo è che tutti gli sprite devono essere, nello stesso momento, della stessa grandezza. È però possibile produrre degli sprite che siano apparentemente di due colori, ed è anche possibile mantenere sulla stessa schermata sprite di differente grandezza e fattore di ingrandimento. Ciò può essere ottenuto mediante il cambiamento delle tabelle degli sprite in condizione d'interruzione. Si noti che, quando si accede al VDP durante un'interruzione, è essenziale che tale interruzione non avvenga mentre il VDP è accessibile dal programma principale. Ciò si ottiene ponendo tutti gli accessi al VDP del programma principale dopo un'istruzione "HALT" dello Z-80 oppure segnalando con un flag che è in corso un accesso al VDP.

## Sprite di due colori

Variando nella tabella degli attributi degli sprite gli elementi in relazione con il nome ed il colore degli sprite ad ogni scansione video, è possibile produrre sprite che siano in apparenza di due colori. Si suggerisce di seguire il metodo che segue:
Prima di tutto definire in due modi gli sprite: una definizione per ciascun colore. Le definizioni devono risiedere nella VRAM una di seguito all'altra, in modo che le definizioni 0 e 1 definiscano un primo sprite multicolore, la definizione 2 e 3 un secondo e così via.
In secondo luogo riservare una zona di memoria per una tabella di scambio. Ciascun elemento di quest'ultima fa riferimento ad un codice di colore per le definizioni di sprite di ordine dispari (nel suo semibyte di ordine maggiore); nel semibyte di ordine inferiore trova posto il codice del colore per le definizioni di ordine pari. Questa tabella va aggiornata ogni volta che viene definito un nuovo sprite. Infine è necessario scrivere una procedura guidata da interruzione per cambiare da una definizione all'altra (e quindi da un colore all'altro) ad ogni interruzione dovuta alla scansione video. La routine che segue illustra quanto detto:

```
SPSWTB:   DEFS 32          ;TABELLA DI SCAMBIO
                           ;DEI COLORI
SPSWIT:   LD HL,SATTAB     ;IND. BASE PER TABELLA ATTR.
                           ;SPRITE IN HL
          LD DE,SPSWTB     ;IND. TABELLA SCAMBIO
                           ;COLORI IN DE
          LD B,32          ;MASS. NUM. ELEMENTI
                           ;DA SCAMBIARE
SPSWLP:   CALL 004AH       ;LEGGE POS. VERT. SPRITE
          CP 0D0H          ;È IL SEGNALE FINE SPRITE?
          RET Z            ;SE È COSÌ FINE
          INC HL           ;SE NO INCREMENTA PUNT.
                           ;TABELLA ATTR. SPRITE
          INC HL           ;DUE VOLTE PER PUNTARE
                           ;NOME SPRITE
          CALL 004AH       ;LEGGI NUM. MODELLO
          XOR 1            ;MODELLO DIVENTA +0 —1
          LD C,A           ;SALVA IL NUOVO NUM.
                           ;MODELLO
          CALL 004DH       ;E LO SCRIVE NEGLI ATTR.
                           ;DELLO SPRITE
          INC HL           ;PUNTA AL TAG DELLO SPRITE
```

```
         LD A,(DE)          ;PRENDE I COLORI
                            ;DA SCAMBIARE
         AND A              ;PULISCE IL FLAG DI RIPORTO
         RR C               ;RUOTA BIT 0 DEL NUM. MOD.
                            ;NEL BIT DI RIPORTO
         JR C,SKP           ;SE MODELLO È DISPARI NON...
         SLA A              ;SPOSTARE IL SEMIBYTE ALTO
                            ;DEL COLORE
         SLA A
         SLA A
         SLA A              ;NEL SEMIBYTE BASSO
SKP:     AND 15             ;MASCHERA IL SEMIBYTE ALTO
         LD C,A             ;MEMORIZZA NUOVO COD.
                            ;COLORE IN C
         CALL 004AH         ;LEGGE VECCHIO TAG
         AND 80H            ;MASCHERA BIT "EARLY CLOCK"
         OR C               ;SOMMA NUOVO COLORE
         CALL 004DH         ;SCRIVE NUOVO TAG
                            ;NEGLI ATTR. SPRITE
         INC HL             ;PUNTA PROSSIMO ELEMENTO
                            ;TAB. ATTR. SPRITE
         INC DE             ;PUNTA PROSSIMO ELEM.
                            ;TABELLA SCAMBIO COLORI
         DJNZ SPSWLP        ;LO FA 32 VOLTE
         RET
```

In questo modo due sprite di differente colore e definizione condividono la medesima entrata nella tabella degli attributi degli sprite, ciascuno dei quali viene visualizzato ad ogni seconda scansione del video televisivo. Può succedere che a causa di ciò insorga un leggero fenomeno di sfarfallio ("flicker"): ma in circostanze normali la persistenza di un tubo catodico è sufficientemente grande da rendere il suddetto fenomeno insignificante.

## Sprite di differente dimensione: scambio dei registri del VDP

Per mantenere sulla stessa schermata sprite di differente grandezza o fattore d'ingrandimento è necessario disporre contemporaneamente nella VRAM di due tabelle degli attributi degli sprite - una per ciascuna grandezza o fattore d'ingrandimento. Poi si fa in modo che il VDP acceda a una e all'altra tra due successive scansioni video.
Nelle modalità multicolore e Grafica I di solito è possibile progettare una disposizione della memoria VRAM in modo da ottenere quanto desiderato senza sovrapporsi agli altri sottoblocchi del VDP. Nella modalità Grafi-

ca II, però, è necessario sovrapporre la seconda tabella degli attributi su una delle altre tabelle. La cosa più conveniente è "rubare", a questo scopo, gli ultimi 16 byte delle definizioni degli sprite: la mappa di memoria che ne risulta viene illustrata nel seguito:

0000H Tabella di generazione dei caratteri
1800H Tabella dei nomi dei modelli
1B00H Tabella degli attributi degli sprite (numero 1)
2000H Tabella dei colori dei caratteri
3800H Tabella di generazione dei modelli degli sprite
3F80H Tabella degli attributi degli sprite (numero 2)

Avendo in questo modo progettato una mappa della memoria adatta allo scopo, è cosa semplice scrivere una routine d'interruzione che scambi i bit di grandezza e fattore d'ingrandimento nel registro 1, e i bit dell'indirizzo di base della tabella degli attributi degli sprite che si trovano nel registro 5. La procedura che segue illustra quanto detto, disponendo di due tabelle degli attributi: una per gli sprite 16 * 16 non ingranditi, l'altra per quelli 16 * 16 ingranditi:

```
         VAL1A EQU 0E2H    ;VALORE DEL REG. 1 PER
                           ;SPRITE 16*16 NON INGRANDITI
         VAL1B EQU 0E3H    ;COME SOPRA PER SPRITE
                           ;INGRANDITI
         VAL5A EQU 036H    ;VALORE REG. 5
                           ;PER IMPOSTARE INDIRIZZO
                           ;TAB. ATTR. A 1B00H
         VAL5B EQU 07FH    ;IMPOSTA TABELLA ATTR.
                           ;A 3F80H
CTR:     DEFB 0            ;IMPOSTA IL CONTATORE
SPATSW:  LD A,(CTR)        ;PRENDE IL CONTATORE
                           ;DI SCAMBIO
         INC A             ;LO INCREMENTA
         LD (CTR),A        ;LO ASSEGNA NUOVAMENTE
         AND 1             ;CONTROLLA BIT 0
         JR NZ,UNMAG       ;SE VALE 1 ATTIVA SPRITE
                           ;NON INGRAND.
         LD B,VAL1B        ;VALORE PER REG. 1
                           ;(SPRITE INGRAND.)
         LD C,1            ;NUM. REGISTRO
         CALL 0047H        ;SCRIVE B NEL REGISTRO C
         LD B,VAL5B        ;VALORE PER REG. 5,
                           ;ATTRIBUTI SPRITE IN 1B00H
```

```
        LD C,5
        CALL 0047H
        RET
UNMAG:  LD B,VAL1A
        LD C,1
        CALL 0047H
        LD B,VAL5A
        LD C,5
        CALL 0047H
        RET
```

La tecnica appena illustrata non solo ci consente di avere contemporaneamente sullo schermo sprite di due differenti grandezze, ma, poiché si hanno a disposizione due tabelle scambiabili di attributi degli sprite, si possono anche avere sullo schermo il doppio dei normali sprite, fino a un massimo di 64. Poiché di volta in volta è attiva una sola tabella degli attributi, possiamo avere fino a 8 sprite sulla stessa linea orizzontale, sempre con un massimo di quattro per ciascuna tabella.
Si deve però ricordare che nei sistemi video dove la persistenza è particolarmente breve, usando le tecniche illustrate si introdurrebbe un "flicker" inaccettabile - sfortunamente non completamente eliminabile.

## Accesso rapido al VDP: come evitare i problemi di temporizzazione

Sebbene il metodo di accedere alla VRAM tramite il VDP abbia il vantaggio di non occupare del prezioso spazio d'indirizzamento della CPU, ha lo svantaggio di essere lento. Questa situazione non è mai completamente evitabile, sebbene i suoi effetti più gravosi possano di solito essere eliminati. Per quanto concerne l'accesso alla VRAM c'è una regola fondamentale, che dovete sforzarvi di seguire quanto più possibile:

USATE IL REGISTRO D'INDIRIZZAMENTO DEL VDP E ACCEDETE SEQUENZIALMENTE ALLA VRAM.

Poiché l'accesso sequenziale alla VRAM richiede un solo trasferimento di dati, mentre quello non sequenziale ne richiede due, in quanto il registro d'indirizzamento va rimpostato ogni volta, il primo è considerevolmente più rapido. Per esempio, quando si scrive una routine per fare uno "scroll" dello schermo è più vantaggioso leggere l'intero schermo (la tabella dei nomi) in un'area temporanea della RAM della CPU per poi riscriverlo per intero.
Usando questa tecnica lo "scroll" può avvenire in un tempo di molto infe-

riore a 1/50 di secondo, mentre una routine analoga che facesse uso dell'accesso non sequenziale richiederebbe un tempo di poco inferiore a 3/50 di secondo.
Se è necessario manipolare di volta in volta grosse regioni della VRAM, è conveniente leggere i dati nella RAM di sistema ed elaborarli in questa sede, prima di ricaricarli nella VRAM stessa in modo sequenziale.
Facendo riferimento alla figura 6.2 si può vedere come l'accesso al VDP è molto più rapido durante i 4,300 microsecondi che seguono immediatamente ogni interruzione dovuta al "flyback" della scansione video. Nei casi in cui la temporizzazione è estremamente critica è quindi vantaggioso effettuare tutti o la maggior parte degli accessi alla VRAM nella routine attivata dall'interruzione.
Riassumendo, per ottenere un rapido accesso alla VRAM, si deve procedere ad ottimizzare quanto segue:
Prima di tutto limitare quanto possibile l'accesso alla VRAM. Tenete una copia dei dati di VRAM che è probabile dobbiate usare nella RAM indirizzabile dalla CPU.
In secondo luogo dovete, quanto più possibile, accedere sequenzialmente alla VRAM: l'accesso non sequenziale è infatti dispendioso in termini di tempo e, se ben si pensa, il più delle volte superfluo.
Infine se proprio il tempo è disperatamente stretto, tentate di accedere alla VRAM durante l'interruzione dovuta alla scansione video, poiché in questo periodo l'accesso alla VRAM non è mai rallentato, in quanto il VDP finisce tutte le attività precedentemente in corso.

Capitolo 7

# IL GENERATORE PROGRAMMABILE DI SUONI

## I registri per i dati

Il generatore di suoni scelto per il sistema MSX è il General Instruments AY-3-8910 (o equivalente). Questo integrato è già stato brevemente discusso nel Capitolo 5. Il dispositivo contiene 16 registri a lettura/scrittura che consentono all'utente di produrre suoni e rumore "bianco" per uno qualsiasi dei tre canali separati. I registri per i dati dell'integrato per la generazione del suono sono:

Registro 0: Regolazione accurata del periodo del suono per il canale A.
Registro 1: Regolazione di massima del periodo del suono per il canale A.
Registro 2: Regolazione accurata del periodo del suono per il canale B.
Registro 3: Regolazione di massima del periodo del suono per il canale B.
Registro 4: Regolazione accurata del canale C.
Registro 5: Regolazione di massima canale C.
Registro 6: Periodo del segnale rumore.
Registro 7: Abilitazione e direzione dell'I/O.
Registro 8: Ampiezza e forma d'onda per il canale A.
Registro 9: Ampiezza e forma d'onda per il canale B.
Registro 10: Ampiezza e forma d'onda per il canale C.
Registro 11: Regolazione fine del periodo dell'onda.
Registro 12: Regolazione del periodo dell'onda.
Registro 13: Forma dell'inviluppo.
Registro 14: Dato memorizzato per la porta A.
Registro 15: Dato memorizzato per la porta B.

**I generatori di suono (registri 0..5)**
Ciascun canale ha associati due registri per il periodo del suono che devono generare. Questi determinano il periodo del suono che deve essere generato in unità di 8 microsecondi. I registri per la regolazione accurata con-

tengono gli 8 bit meno significativi del periodo selezionato, mentre i registri di regolazione di massima contengono i quattro bit di ordine superiore. Per far sì che l'integrato emetta suono da un determinato canale deve essere posto a zero il bit appropriato del registro 7 (di abilitazione).

**Il generatore di rumore (registro 6)**
Si tratta del registro che controlla il singolo generatore di rumore pseudocasuale di cui dispone l'integrato in questione. L'uscita da questo generatore può essere mischiata a uno degli altri tre canali, quando ciò sia richiesto, ponendo a zero il bit appropriato del registro 7. Il periodo del generatore di rumore è determinato dai cinque bit meno significativi di questo registro.

**Il registro di abilitazione (registro 7)**
Il registro 7 specifica se l'uscita dei tre canali debba comprendere il suono o il rumore, oppure entrambi. Determina inoltre se le due porte d'ingresso/uscita sono usate in ingresso o in uscita. Il significato dei bit è il seguente:

Bit 0: Se posto a 1 disabilita il suono in uscita dal canale A.
Bit 1: Se posto a 1 disabilita il suono in uscita dal canale B.
Bit 2: Se posto a 1 disabilita il suono in uscita dal canale C.
Bit 3: Disabilitazione del rumore dal canale A.
Bit 4: Disabilitazione del rumore dal canale B.
Bit 5: Disabilitazione del rumore dal canale C.
Bit 6: Se posto a 1 determina l'uso della porta A con modalità di uscita.
Bit 7: Se posto a 0 determina l'uso della porta B con modalità di ingresso.

Si noti che nel sistema MSX le porte d'ingresso/uscita sono usate per leggere l'ingresso dal joystick, e quindi vanno sempre impostate con modalità d'ingresso.

**Controllo d'ampiezza (registri 8..10)**
Ciascun canale ha il suo proprio registro di controllo dell'ampiezza. Il bit 4 di questo registro specifica se va usata una definizione via hardware per il canale in questione. Ponendo a 1 questo bit, infatti, si pone l'ampiezza sonora sotto il controllo del generatore hardware d'inviluppo. Viceversa se il bit è posto a 0, l'ampiezza viene determinata via software usando i bit da 0 a 3, dove il valore 0 indica il minimo di volume mentre 15 il massimo.

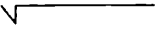
**Generatore d'inviluppo (registri 11..13)**
L'integrato per la generazione di suoni ha un solo generatore d'inviluppo hardware che può essere usato da uno o da tutti e tre i canali, secondo quanto viene indicato nel registro di controllo d'ampiezza.

| VALORE | FORMA |
|---|---|
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |

*Fig. 7.1 - Forme d'inviluppo.*

I bit da 0 a 3 del registro di controllo 13 determinano la forma dell'inviluppo in un modo piuttosto insolito. Nella figura 7.1 vengono dati i valori necessari per generare via hardware le otto forme d'inviluppo disponibili. Ogni altro valore implica la duplicazione delle forme 9 o 15.
La lunghezza di ogni rampa, sia verso l'alto sia verso il basso, viene impostata tramite il periodo dell'onda.
Il periodo dell'onda è un valore su 16 bit, che viene memorizzato dai registri 11 (LSB) e 12 (MSB). Il periodo viene indicato in unità di 128 microsecondi ed indica il tempo da un passo all'altro della rampa. Se una rampa ha 16 passi (variando il volume da 0 a 15) il tempo totale della rampa stessa è di 1024 microsecondi per il periodo dell'onda. Quindi il periodo dell'inviluppo dell'onda indica, approssimativamente, la lunghezza della rampa in millisecondi.

**Le porte di ingresso/uscita (registri 14 e 15)**
Poiché i registri d'ingresso/uscita sono usati dal sistema MSX per leggere dalle porte del joystick, è meglio non tentare di utilizzarli altrimenti: inol-

tre la tecnica con cui sono stati bufferizzati è piuttosto strana, ma risulta utile nella fase di configurazione del MSX.

**Note e periodi del suono**

Il periodo del suono richiesto può essere calcolato a partire dalla frequenza richiesta con la seguente formula:

Periodo = 125000 / Frequenza

Inoltre la frequenza di ogni nota nella scala temperata nell'intero intervallo di otto ottave viene calcolata, secondo lo Standard Internazionale della nota la. come segue:

Frequenza = 440 * (2^(Ottava+(N—10)/12))

Dove:

Ottava è il numero di ottava. 0 è l'ottava che contiene il DO centrale, —1 è l'ottava inferiore seguente, 1 quella superiore e così via.

N è il numero della nota: 1 è il DO, 2 è il DO#, 3 è il RE ecc.

Poiché il periodo è un valore intero, si deduce che i valori calcolati dalle formule sopra riportate non producono esattamente la frequenza desiderata. Gli errori sono però molto piccoli e di solito non si notano.
La tabella che segue riporta i valori per l'Ottava 0 dedotti dalle formule e li mette in relazione con un fattore di errore, calcolato come percentuale della frequenza richiesta:

| Nota | Frequenza | Periodo | Errore |
|---|---|---|---|
| DO | 261.626 | 478 | 0.046% |
| DO# | 277.183 | 451 | 0.007% |
| RE | 293.665 | 426 | 0.081% |
| RE# | 311.127 | 402 | 0.058% |
| MI | 329.628 | 379 | 0.057% |
| FA | 349.228 | 358 | 0.019% |
| FA# | 369.994 | 338 | 0.046% |
| SOL | 391.995 | 319 | 0.037% |
| SOL# | 415.305 | 301 | 0.005% |
| LA | 440.000 | 284 | 0.032% |
| LA# | 466.164 | 268 | 0.055% |
| SI | 493.883 | 253 | 0.038% |

Date le formule riportate in precedenza, la procedura BASIC che segue calcola il periodo del suono per le note dell'intero intervallo di otto ottave in base ad una stringa d'ingresso nel formato 'ON'(Ottava, Nota). Però, per semplificare l'analisi lessicale della stringa in ingresso, i valori dell'ottava variano tra 0 e 7 invece che tra —3 e 4.
La parte più a destra della stringa indica la nota (da notare la denominazione inglese delle note: A = la, B = si, C = do, D = re, E = mi, F = fa, G = sol): usa due caratteri quando la nota è seguita dal segno # (diesis):

```
  10 INPUT A$
  20 GOSUB 1000
  30 PRINT A
  40 GOTO 10
1000 O = VAL(LEFT$(A$,1))
1010 A$ = MID$(A$,2)
1020 FOR NC = 1 TO 12
1030 READ B$:IF B$=A$ THEN NO=NC
1040 NEXT C
1050 RESTORE
1060 P = 125000 / (440*(2^((O—3)+(NO—10)/12))
1070 A=INT(P)
1080 RETURN
1090 DATA C,C#,D,D#,E,F,F#,G,G#,A,A#,B
```

Se la procedura appena vista viene espansa, diventa semplice ottenere un programma che fornisca i periodi dei suoni in modo che possano essere usati in routine "musicali" scritte in codice macchina.

## Accesso al PSG in ambiente MSX

Poiché i progettatori del MSX si riservano il diritto di apportare modifiche all'hardware del sistema stesso, garantendo la compatibilità del software, è possibile programmare il PSG solo attraverso le chiamate al sistema operativo che vengono messe a disposizione nella ROM di sistema. Segue una descrizione di queste routine:

| INDIRIZZO | FUNZIONE |
|---|---|
| 0090H | Questa routine inizializza il PSG; non richiede parametri e può modificare tutti i registri. |
| 0093H | Questa routine scrive il contenuto del registro E nel registro del PSG specificato per mezzo dell'accumulato- |

re; non ritorna alcun valore e lascia inalterati tutti i registri.

0096H Questa routine legge un dato dal registro specificato dall'accumulatore, ritornando nell'accumulatore stesso il dato letto e non modificando gli altri registri.

## Programmazione del PSG

La programmazine del PSG consiste in una serie di scritture nei suoi registri per specificare il periodo del suono per un dato canale, il periodo del rumore, se richiesto, una qualsiasi forma d'inviluppo che l'utente intenda utilizzare, il periodo dell'onda e (se non devono essere usate forme d'inviluppo particolari) l'ampiezza per il canale prescelto. Queste azioni andrebbero effettuate con il canale prescelto in condizione di disabilitazione (che si ottiene ponendo a 1 il bit ENABLE del registro 7). Consideriamo, come esempio di quanto si è detto, la seguente routine, che provoca la generazione di un suono DO centrale da parte del canale A, con la forma d'inviluppo d'ampiezza 8:

```
MIDC:   LD A,7
        CALL 0096H      ;LEGGE REGISTRO
                        ;ABILITAZIONE
        OR 9            ;DISABILITA SUONO E RUMORE
                        ;PER CANALE A
        LD E,A
        LD A,7
        CALL 0093H      ;RISCRIVE IL REGISTRO
        LD E,1
        LD A,1
        CALL 0093H      ;IMPOSTA LA REGOLAZIONE
                        ;DEL SUONO CANALE A
        LD E,0DEH
        LD A,0
        CALL 0093H      ;IMPOSTA LA REGOLAZIONE
                        ;ACCURATA PER DO CENTRALE
        LD E,8
        LD A,13
        CALL 0093H      ;IMPOSTA FORMA ONDA
        LD E,15
        LD A,11
        CALL 0093H      ;PERIODO DEL SUONO
                        ;(REG. ACCURATA)
```

```
LD E,0
LD A,12
CALL 0093H          ;PERIODO DEL SUONO
                    ;(REG. DI MASSIMA)
LD E,16
LD A,8
CALL 0093H          ;ABILITA CONTROLLO ONDA
                    ;DEL CANALE A
LD A,7
CALL 0096H          ;LEGGE REGISTRO
                    ;ABILITAZIONE
AND 0FEH            ;PONE A 0 FLAG
                    ;DISABILITAZIONE CANALE A
LD E,A
LD A,7
CALL 0093H          ;SCRIVE NUOVI VALORI
                    ;ABILITAZIONE
RET
```

Sebbene questa routine sia abbastanza banale, illustra efficacemente le tecniche di programmazione del PSG. Nel caso di un integrato come l'AY-3-8910, dotato di enormi potenzialità, l'unico modo per sfruttarlo appieno consiste nella sperimentazione. In questa prospettiva si è dedicato il resto del capitolo alla presentazione di una serie di esempi che hanno lo scopo di familiarizzare ulteriormente il lettore con questa componente del sistema MSX.

## Musica su tre canali: il computer artista

Il programma che viene presentato è guidato da interruzione in modo da consentire la generazione di suono da parte dei tre canali musicali, indipendentemente dalla contemporanea esecuzione di un altro programma. I dati per i tre canali musicali devono essere memorizzati agli indirizzi indicati come C1DAT, C2DAT e C3DAT come una serie di elementi di tre byte. Il primo byte di ciascun elemento è il valore di regolazione accurata per il periodo della nota, il byte due è per la sua regolazione di massima mentre il byte tre indica la durata della nota in unità di 20 millisecondi. La fine del brano viene indicata scrivendo 255 nel byte di regolazione di massima dell'ultimo elemento di dati per il canale A. Mentre il programma è in esecuzione, e quindi viene eseguito il brano musicale, il volume di ciascun canale può essere modificato scrivendo agli indirizzi indicati dai

simboli VOL1, VOL2 e VOL3. Se desiderate fare delle prove con il generatore hardware di forma d'inviluppo potete attivarlo con il comando BASIC SOUND e quindi scrivere il valore 16 all'indirizzo del volume del canale voluto.

```
        INTHOK  EQU 0F9DFH ;INDIRIZZO "AGGANCIO"
                           ;D'INTERRUZIONE
        PSGINI  EQU 00090H ;ROUTINE INIZIALIZZAZIONE
                           ;PSG
        WRTPSG  EQU 00093H ;PER SCRIVERE DATI NEL PSG
        RDPSG   EQU 00096H ;PER LEGGERE DATI DAL PSG
        C1DAT   EQU 0B000H ;INIZIO DATI CANALI 1
        C2DAT   EQU 0B400H ;INIZIO DATI CANALE 2
        C3DAT   EQU 0B800H ;INIZIO DATI CANALE 3: TUTTI
                           ;POSSONO ESSERE MODIFICATI
                           ;SE RICHIESTO
        ORG 9000H
START:  LD HL,C1DAT
        LD (C1PTR),HL
        LD HL,C2DAT
        LD (C2PTR),HL
        LD HL,C3DAT
        LD (C3PTR),HL      ;IMPOSTA PUNTATORI AI DATI
        LD A,1
        LD (C1CTR),A
        LD (C2CTR),A
        LD (C3CTR),A       ;IMPOSTA I CONTATORI
                           ;DI DURATA DELLE NOTE
        CALL PSGINI        ;INIZIALIZZA IL PSG
        LD A,LOW MUSROT    ;PRENDE INDIRIZZO BASSO
                           ;DELLA ROUTINE MUSICALE
        LD (INTHOK+1),A    ;LO SCRIVE NELL'"AGGANCIO"
                           ;D'INTERRUZIONE
        LD A,HIGH MUSROT
        LD (INTHOK+2),A    ;SCRIVE IND. ALTO COME SOPRA
        LD A,0C3H          ;PRENDE ISTRUZIONE 'JP'
                           ;DELLO Z-80
        LD (INTHOK),A      ;CARICA NELL'"AGGANCIO"
                           ;D'INTERRUZIONE
        RET                ;ORA L'INTERRUZIONE ABILITA
                           ;LA ROUTINE MUSICALE
C1PTR:  DEFW 0
```

```
C2PTR:    DEFW 0
C3PTR:    DEFW 0
C1CTR:    DEFB 0
C2CTR:    DEFB 0
C3CTR:    DEFB 0
VOL1:     DEFB 15
VOL2:     DEFB 15
VOL3:     DEFB 15             ;IMPOSTA TABELLA
                              ;DELLE VARIABILI
MUSROT:   PUSH AF             ;SALVA STATO VDP
          LD A,(C1CTR)        ;INIZIO ROUTINE MUSICALE
          DEC A               ;DECREMENTA DURATA
                              ;DELLA NOTA
          LD (C1CTR),A
          OR A                ;LA DURATA È 0: CIOÈ LA NOTA
                              ;È FINITA
          JR NZ,CHANB         ;SE DIVERSA DA 0 VA
                              ;SUL CANALE 2
          LD A,7              ;SE NOTA È FINITA
          CALL RDPSG
          OR 1
          LD E,A
          LD A,7
          CALL WRTPSG         ;ALLORA SPEGNE CANALE 1
          LD IX, (C1PTR)      ;PORTA PUNTATORE DATI IN IX
          LD E,(IX+0)         ;PRENDE VALORE
                              ;REGOLAZIONE ACCURATA
          LD A,0
          CALL WRTPSG         ;LO SCRIVE NEL REGISTRO 0
                              ;DEL PSG
          LD A,(IX+1)         ;PRENDE VALORE
                              ;REGOLAZIONE DI MASSIMA
          CP 255              ;SE UGUALE AL DELIMITATORE
                              ;DEL BRANO
          JR Z,START          ;RINIZIALIZZA
          LD E,A
          LD A,1
          CALL WRTPSG         ;ALTRIMENTI LO SCRIVE
                              ;NEL REG. 1
          LD A,(IX+2)         ;IMPOSTA CONTATORE DURATA
                              ;DELLA NOTA
          LD (C1CTR),A
```

```
        LD A,(VOL1)         ;PRENDE VOLUME PER CANALE A
        LD E,A
        LD A,8
        CALL WRTPSG         ;SCRIVE VOLUME
                            ;CANALE 1
        LD A,7
        CALL RDPSG
        AND 0FEH
        LD E,A
        LD A,7
        CALL WRTPSG         ;RIABILITA PROD. SUONO
                            ;DAL CANALE 1
        INC IX
        INC IX
        INC IX              ;PUNTA PROSSIMA NOTA
        LD (C1PTR),IX       ;SALVA PUNTATORE
CHANB:  LD A,(C2CTR)        ;CANALE B
        DEC A               ;DECREMENTA DURATA
                            ;DELLA NOTA
        LD (C2CTR),A
        OR A                ;LA DURATA È 0: CIOÈ LA NOTA
                            ;È FINITA
        JR NZ,CHANC         ;SE DIVERSA DA 0 VA
                            ;SUL CANALE 3
        LD A,7              ;SE NOTA È FINITA
        CALL RDPSG
        OR 2
        LD E,A
        LD A,7
        CALL WRTPSG         ;ALLORA SPEGNE CANALE 2
        LD IX, (C2PTR)      ;PORTA PUNTATORE DATI IN IX
        LD E,(IX+0)         ;PRENDE VALORE
                            ;REGOLAZIONE ACCURATA
        LD A,2
        CALL WRTPSG         ;LO SCRIVE NEL REGISTRO 0
                            ;DEL PSG
        LD E,(IX+1)         ;PRENDE VALORE
                            ;REGOLAZIONE DI MASSIMA
        LD A,3
        CALL WRTPSG
        LD A,(VOL2)
        LD E,A
```

```
        LD A,9
        CALL WRTPSG         ;ALTRIMENTI LO SCRIVE
                            ;NEL REG. 1
        LD A,7
        CALL RDP5G
        AND 0FDH
        LD E,A
        LD A,7
        CALL WRT PSG
        LD A,(IX+2)         ;IMPOSTA CONTATORE DURATA
                            ;DELLA NOTA
        LD (C2CTR),A
        INC IX
        INC IX
        INC IX
        LD (C2PTR),IX
CHANC:  LD A,(C3CTR)        ;CANALE C
        DEC A               ;DECREMENTA DURATA
                            ;DELLA NOTA
        LD (C3CTR),A
        OR A                ;LA DURATA È 0: CIOÈ LA NOTA
                            ;È FINITA
        JR NZ,ENDMUS
        LD A,7              ;SE NOTA È FINITA
        CALL RDPSG
        OR 4
        LD E,A
        LD A,7
        CALL WRTPSG         ;ALLORA SPEGNI CANALE 3
        LD IX, (C3PTR)      ;PORTA PUNTATORE DATI IN IX
        LD E,(IX+0)         ;PRENDE VALORE
                            ;REGOLAZIONE ACCURATA
        LD A,4
        CALL WRTPSG         ;LO SCRIVE NEL REGISTRO 0
                            ;DEL PSG
        LD E,(IX+1)         ;PRENDE VALORE
                            ;REGOLAZIONE DI MASSIMA
        LD A,5
        CALL WRTPSG
        LD A,(IX+2)         ;IMPOSTA CONTATORE DURATA
                            ;DELLA NOTA
        LD (C3CTR),A
```

```
                LD A,(VOL3)
                LD E,A
                LD A,10
                CALL WRTPSG          ;ALTRIMENTI LO SCRIVE
                                     ;NEL REG. 1
                LD A,7
                CALL RDPSG
                AND 0FBH
                LD E,A
                LD A,7
                CALL WRTPSG
                INC IX
                INC IX
                INC IX
                LD (C3PTR),IX
ENDMUS:         POP AF               ;RIPRISTINA STATO VDP
                RET
```

## Effetti sonori con l'AY-3-8910

La produzione di effetti sonori è un campo applicativo dove i risultati si possono ottenere soltanto con la sperimentazione. Infatti se è vero che il PSG è in grado di produrre praticamente qualsiasi suono voi desideriate, è altrettanto vero che potreste impiegare ore provando a impostare con diversi valori i suoi registri prima di trovare una soluzione soddisfacente. Per far ciò il modo probabilmente migliore è il comando BASIC SOUND con il quale potete velocemente e con semplicità cambiare il contenuto di uno qualsiasi dei registri del PSG. Una volta trovati i valori desiderati per i suoi registri, si può poi codificare la necessaria routine in codice macchina per caricarli nei registri stessi in modo veloce ed efficiente.
Per esempio il seguente programma BASIC produce il suono di uno sparo:

```
10 SOUND 6,15:SOUND 7,7
20 SOUND 8,16:SOUND 9,16:SOUND 10,16
30 SOUND 11,0:SOUND 12,16:SOUND 13,0
```

Il codice macchina Z-80 equivalente è il seguente:

```
GUN:            LD B,6
                LD HL,GUNTBL
GUNLP:          LD E,(HL)
```

```
          LD A,B
          CALL 0093H
          INC HL
          INC B
          LD A,B
          CP 14
          LR NZ,GUNLP
          RET
GUNTBL:   DEFB 15,7,16,16,16,0,16,0
```

## Generazione di suoni via software: la porta di un bit

Oltre alla generazioni di suoni per mezzo dell'hardware del PSG, il sistema MSX riserva un bit della porta C del PPI 8255 per la generazione di suoni via software. Una chiamata al sistema operativo all'indirizzo 0135H accetta un valore in ingresso nell'accumulatore: se il valore è 0 il bit sonoro della porta in questione viene posto a 0, e viceversa se il valore è 1. Cambiando rapidamente il valore di questo bit si può ottenere la generazione di suono via software; provate le seguente routine:

```
SFTSND:   LD A,0
          CALL 135H
          LD A,1
          CALL 135H
          JR SFTSND
```

Facendola eseguire potrete udire un suono acuto. Potete provare a sentire cosa accade nel cambiare il valore del suddetto bit con diverse frequenze.

**Capitolo 8**

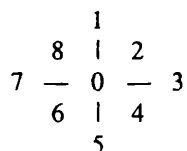
# INGRESSO/USCITA: LA FINESTRA DEL COMPUTER SUL MONDO

Il sistema MSX fornisce un insieme piuttosto vasto di procedure per effettuare funzioni come la lettura e scansione della tastiera, la lettura dell'ingresso da joystick e delle tavolette per i giochi e la visualizzazione su schermo. In questo capitolo si prende dapprima in considerazione l'ingresso da joytick e poi l'ingresso/uscita in generale dalla console.

## I/O per i giochi: joystick e tavolette tattili

Il sistema MSX prevede un massimo di due joystick, e considera come un terzo joystick le frecce direzionali per il cursore e la barra spaziatrice. Le seguenti procedure vengono usate per leggere lo stato del joystick e per rivelare la pressione del suo pulsante.

0D5H Legge lo stato corrente del joystick specificato ponendo il valore letto nell'accumulatore (il numero di joystick varia da 0 a 2, dove 0 indica le frecce del cursore). Il valore ritornato varia tra 0 e 8. Si veda il diagramma che segue. Il valore 0 significa che il joystick è centrato. Questa procedura può modificare tutti i registri.

```
        1
    8   |   2
7   —   0   —   3
    6   |   4
        5
```

0D8H Questa procedura rivela la pressione del pulsante del joystick specificato nell'accumulatore. Il valore ritornato è 0 se il pulsante non è premuto, 255 viceversa. Viene modificata solo la coppia di registri AF. (Quando si usa il valore 0 per identificare il joystick, la routine in questione rivela la pressione della barra spaziatrice).

Le porte per i joystick sono adatte anche ai paddle per i giochi. È possibile, nel sistema MSX, connetterne fino a 6 per ogni porta per joystick, per un totale di 12. Essi vengono numerato da 1 a 12. I paddle con numero dispari vengono connessi alla porta joystick 1, quelli pari alla porta 2.

0DEH Questa procedura legge il paddle il cui numero è specificato nell'accumulatore, ritornando, nell'accumulatore stesso, un valore tra 0 e 255 dipendente dalla posizione del paddle specificato. Può modificare tutti i registri.

Come ultima cosa di questa sezione, diciamo che alle porte joystick possono essere connesse tavolette tattili del tipo NEC PC-6051. La routine che segue viene usata per leggere lo stato della tavoletta tattile:

0DBH Questa procedura accetta un identificatore di tavoletta nell'accumulatore, e ritorna un valore letto dalla tavoletta tattile stessa, modificando tutti i registri. Gli identificatori e i valori ritornati sono i seguenti:

| *ID* | *Valore ritornato* |
|---|---|
| 0 | 255 se la tavoletta nella porta 1 è premuta, 0 viceversa. |
| 1 | Ritorna la coordinata X del punto di pressione sulla tavoletta connessa alla porta 1. |
| 2 | Come sopra, la coordinata Y. |
| 3 | Ritorna 255 se il selettore sulla tavoletta 1 è premuto, 0 altrimenti. |
| 4-7 | Come sopra per la tavoletta tattile connessa alla porta 2. |

## Ingresso/uscita da console

Il sistema operativo fornisce le seguenti procedure per accedere alla tastiera e al video:

09CH Controlla lo stato del buffer della tastiera. Ritorna con il flag

zero posto a 1 se il buffer è vuoto, e modifica la coppia di registri AF.

| | |
|---|---|
| 09FH | Preleva un carattere dal buffer d'ingresso, o, se il buffer è vuoto, attende che un carattere venga introdotto. Ritorna il codice ASCII del carattere stesso nell'accumulatore e modifica la coppia AF. |
| 0A2H | Visualizza un carattere, il cui codice viene preventivamente impostato nell'accumulatore, nella posizione corrente del cursore, lasciando inalterati tutti i registri. |
| 0C0H | Produce un breve impulso sonoro (equivalente al CHR$(7)). Può modificare tutti i registri. |
| 0C3H | Pulisce lo schermo, modificando le coppie di registri AF,BC e DE. |
| 0C6H | Posiziona il cursore dello schermo sulla colonna H, riga L modificando la coppia AF. |
| 0CCH | Cancella la visualizzazione dei tasti funzionali, modificando tutti i registri. |
| 0CFH | Visualizza i tasti funzionali, modificando tutti i registri. |

L'uso di queste procedure è sufficientemente evidente: la costruzione di applicazioni utente non dovrebbe porre quindi particolari problemi, poiché esse rendono disponibili tutte le funzioni necessarie per la gestione dello schermo e della tastiera di console; ad esempio, una semplice routine di ingresso/uscita potrebbe essere costruita così:

```
        LD HL,0
        CALL 0C6H       ;CURSORE IN ALTO A SINISTRA
        CALL 0C3H       ;SCHERMO PULITO
INLP:   CALL 09FH       ;ACCETTA UN CARATTERE
        CALL 0A2H       ;LO STAMPA
        CP 13
        JR NZ,INPL      ;SE NON È <CR> RIPETE
```

Si può anche scrivere una routine per richiedere l'introduzione di una parola chiave che consenta la prosecuzione dell'esecuzione di un programma:

```
PSSWD:  LD HL,PSWORD    ;HL PUNTA ALLA STRINGA
                        ;CONTENENTE LA PAROLA
                        ;CHIAVE
PWLP:   CALL 09FH       ;ACCETTA UN CARATTERE
                        ;DA TASTIERA
        CP(HL)          ;CORRISPONDE ALLA PAROLA
                        ;CHIAVE?
```

```
         JR Z,PWSKP          ;SE SÌ SALTA ALLA FINE
                             ;DEL CICLO
         LD A,(HL)
         CP '&'              ;FINE PAROLA CHIAVE?
         JR Z,GO             ;SE SÌ SALTA AL RESTO
                             ;DEL PROGRAMMA
         JP 0000H            ;ALTRIMENTI RIPARTE
PWSKP:   INC HL              ;INCREMENTA PUNTATORE
                             ;ALLA PAROLA CHIAVE
         CALL 0A2H           ;STAMPA IL CARATTERE
                             ;PRECEDENTE
         JR PWLP             ;CONTROLLA IL PROSSIMO
                             ;CARATTERE
PSWORD:  DEFB 'QUALSIASI PAROLA CHIAVE DESIDERATE
         TERMINATA DA &'
GO:      ;INSERIRE QUI IL RESTO DEL PROGRAMMA
```

## Selezione della partizione

Poiché il PPI 8255 non è posto necessariamente allo stesso indirizzo in due differenti elaboratori MSX, sono state messe a disposizione due procedure per facilitare la lettura e la scrittura della porta A dell'8255 stesso (cioè il registro per la selezione della partizione primaria):

0138H   Legge il valore corrente del registro di selezione della partizione, ritornando il valore nell'accumulatore e lasciando inalterati tutti gli altri registri.

013BH   Scrive il valore dell'accumulatore nel registro di selezione della partizione primaria, lasciando inalterati gli altri registri.

Se quindi volete selezionare le pagine 2 e 3 della partizione 2 lasciando invariate le pagine 0 e 1, per esempio, dovete effettuare le seguenti operazioni:

```
         CALL 138H           ;LEGGE LA SELEZIONE
                             ;DELLA PARTIZIONE
         AND 15              ;MASCHERA I BIT PER LE
                             ;PAGINE 0 E 1
         OR 10100000B        ;MASCHERA PER SELEZIONARE
                             ;PARTIZ. 2 PER PAGINE 2 E 3
         CALL 13BH           ;IMPOSTA IL REGISTRO
                             ;CON IL NUOVO VALORE
```

Esistono altre due routine di sistema operativo che interagiscono con il PPI 8255: la prima si trova alla locazione 0132H, accetta in ingresso il valore dell'accumulatore e se è 0 spegne la luce dei CAPS, altrimenti la accende. La seconda è usata per scandire la tastiera.

## Scansione della tastiera: controllo dei singoli tasti

0141H La routine posta a questo indirizzo scandisce una riga della matrice della tastiera (si veda Figura 8.1) secondo il valore impostato nell'accumulatore (da 0 a 9) e ritorna, nell'accumulatore stesso, un valore tale che il bit corrispondente a ciascuna colonna della matrice, nella riga selezionata, è posto a zero se il tasto corrispondente è premuto. Se nessun tasto è premuto, la routine ritorna 255.

COLONNE

| RIGHE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | + | [ | | ¥ | – | = | 9 | 8 |
| 2 | B | A | – | / | › | ‹ | ] | * |
| 3 | J | I | H | G | F | E | D | C |
| 4 | R | Q | P | O | N | M | L | K |
| 5 | Z | Y | X | W | V | U | T | S |
| 6 | F3 | F2 | F1 | CODE | CAP | GRAPH | CTRL | SHIFT |
| 7 | RETURN | SELECT | BS | STOP | TAB | ESC | FS | F4 |
| 8 | → | ↓ | ↑ | ← | DEL | INS | HOME | SPACE |
| 9 | | | | | | | | |

*Fig. 8.1 - La matrice della tastiera*

Per chiarire quanto detto, considerate la seguente procedura, che scandisce la tastiera sino a quando i tasti 'Z'e 'X'sono premuti contemporaneamente:

```
ZXCHK:   LD A,5              ;SCANDISCE LA RIGA 5
                             ;DELLA MATRICE TASTIERA
         CALL 0141H
         AND 10100000B       ;CONTROLLA CHE I BIT 5 E 7
                             ;SIANO ENTRAMBI A ZERO
         JR NZ,ZXCHK         ;SE NON LO SONO RIPETE
         RET
```

# APPENDICI

## Appendice A

# TABELLA CODICI CARATTERE

Riprodotta per gentile concessione della Toshiba UK Limited.

4 bit più significativi →

↑ 4 bit meno significativi

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | Blank (Space) | Ø | @ | P | ` | p | C | É | á | Ã | | | α | ≡ |
| 1 | | | ! | 1 | A | Q | a | q | ü | æ | í | ā | | | β | ± |
| 2 | | INS | " | 2 | B | R | b | r | é | Æ | ó | Ĩ | | | Γ | ≥ |
| 3 | | | # | 3 | C | S | c | s | â | ô | ú | Ī | | | Π | ≤ |
| 4 | | | $ | 4 | D | T | d | t | ä | ö | ñ | Õ | | | Σ | ⌠ |
| 5 | | | % | 5 | E | U | e | u | à | ò | Ñ | ō | | | σ | ⌡ |
| 6 | | | & | 6 | F | V | f | v | å | û | a | Ũ | | | μ | ÷ |
| 7 | BL | | ' | 7 | G | W | g | w | ç | ù | o | ū | | | Υ | ≈ |
| 8 | BS | SELECT | ( | 8 | H | X | h | x | ê | ÿ | ¿ | Π | | △ | Φ | ° |
| 9 | TAB | | ) | 9 | I | Y | i | y | ë | Ö | ┌ | ij | | ‡ | Θ | • |
| A | LF | | * | : | J | Z | j | z | è | Ü | ┐ | ¾ | | ω | Ω | · |
| B | HOME | ESC | + | ; | K | [ | k | { | ï | ¢ | ½ | ∼ | | | δ | √ |
| C | CLS | → | , | < | L | \ | l | ¦ | î | £ | ¼ | ◊ | | | ∞ | ⁿ |
| D | CR | ← | – | = | M | ] | m | } | ì | ¥ | ¡ | ‰ | | | ϕ | ² |
| E | | ↑ | . | > | N | ^ | n | ~ | Ä | Pt | ≪ | ¶ | | | ∈ | |
| F | | ↓ | / | ? | O | _ | o | Blank (DEL) | Å | ƒ | ≫ | § | | | ∩ | Blank (FF) |

↑ Numeri esadecimali

Numeri esadecimali ←

Appendice B
# TABELLA DEI COLORI

0 .... Trasparente
1 .... Nero
2 .... Verde medio
3 .... Verde chiaro
4 .... Blu scuro
5 .... Blu chiaro
6 .... Rosso scuro
7 .... Ciano
8 .... Rosso medio
9 .... Rosso chiaro
10 .... Giallo scuro
11 .... Giallo chiaro
12 .... Verde scuro
13 .... Rosso Magenta
14 .... Grigio
15 .... Bianco

## Appendice C

# TABELLE DELLA RAM VIDEO

| | Testo 40 col. | Testo 32 col. | HRG | Multicolore |
|---|---|---|---|---|
| Tabella nomi | 0 | 6144 | 6144 | 2048 |
| Tabella modelli | 2048 | 0 | 0 | 0 |
| Tabella colori | — | 8192 | 8192 | — |
| Tabella attributi sprite | — | 6912 | 6912 | 6912 |
| Tabella modelli sprite | — | 14336 | 14336 | 14336 |

Appendice D

# ISTRUZIONI Z-80

I simboli riportati sotto ogni flag hanno i seguenti significati:
R: il flag è aggiornato quale risultato dell'operazione.
0: il flag è posto a 0
1: il flag è posto a 1
C: lo stato del flag di riporto è copiato nel flag H.
La lista delle altre abbreviazioni è riportata alla fine di quest'Appendice.

| | **FLAG** | | | | | |
|---|---|---|---|---|---|---|
| | **S** | **Z** | **H** | **N** | **P** | **C** |
| ADC HL,ss | R | R | R | 0 | R | R |
| ADC A,s | R | R | R | 0 | R | R |
| ADD A,n | R | R | R | 0 | R | R |
| ADD A,r | R | R | R | 0 | R | R |
| ADD A,(HL) | R | R | R | 0 | R | R |
| ADD A,(IX+d) | R | R | R | 0 | R | R |
| ADD A,(IY+d) | R | R | R | 0 | R | R |
| ADD HL,ss | - | - | R | 0 | - | R |
| ADD IX,pp | - | - | R | 0 | - | R |
| ADD IY,rr | - | - | R | 0 | - | R |
| AND s | R | R | 1 | 0 | R | 0 |
| BIT b,(HL) | - | R | 1 | 0 | - | - |
| BIT b(IX+d) | - | R | 1 | 0 | - | - |
| BIT b,(IY+d) | - | R | 1 | 0 | - | - |
| BIT b,r | - | R | 1 | 0 | - | - |
| CALL cc,nn | - | - | - | - | - | - |
| CALL nn | - | - | - | - | - | - |
| CCF | - | - | C | 0 | - | R |
| CP s | R | R | R | 1 | R | R |
| CPD | R | R | R | 1 | R | - |
| CPDR | R | R | R | 1 | R | - |

| | **FLAG** | | | | | |
|---|---|---|---|---|---|---|
| | **S** | **Z** | **H** | **N** | **P** | **C** |
| CPI | R | R | R | 1 | R | - |
| CPIR | R | R | R | 1 | R | - |
| CPL | - | - | 1 | 1 | - | - |
| DAA | R | R | R | - | R | R |
| DEC m | R | R | R | 1 | R | - |
| DEC IX | - | - | - | - | - | - |
| DEC IY | - | - | - | - | - | - |
| DEC ss | - | - | - | - | - | - |
| DI | - | - | - | - | - | - |
| DJNZ e | - | - | - | - | - | - |
| EI | - | - | - | - | - | - |
| EX(SP),HL | - | - | - | - | - | - |
| EX(SP),IX | - | - | - | - | - | - |
| EX(SP),IY | - | - | - | - | - | - |
| EX AF,AF' | R | R | R | R | R | R |
| EX DE,HL | - | - | - | - | - | - |
| EXX | - | - | - | - | - | - |
| HALT | - | - | - | - | - | - |
| IM 0 | - | - | - | - | - | - |
| IM 1 | - | - | - | - | - | - |
| IM 2 | - | - | - | - | - | - |

| Instruction | Flags |
|---|---|
| IN A,(N) | - - - - - - |
| IN R,(C) | R R R 0 R - |
| INC (HL) | R R R 0 R - |
| INC IX | - - - - - - |
| INC(IX+d) | R R R 0 R - |
| INC IY | - - - - - - |
| INC (IY+d) | R R R 0 R - |
| INC r | R R R 0 R - |
| INC ss | - - - - - - |
| IND | - R - 1 - - |
| INDR | - 1 - 1 - - |
| INI | - R - 1 - - |
| INIR | - 1 - 1 - - |
| JP(HL) | - - - - - - |
| JP(IX) | - - - - - - |
| JP(IY) | - - - - - - |
| JP cc,nn | - - - - - - |
| JP nn | - - - - - - |
| JR C,e | - - - - - - |
| JR e | - - - - - - |
| JR NC,e | - - - - - - |
| JR NZ,e | - - - - - - |
| JR Z,e | - - - - - - |
| LD A,(BC) | - - - - - - |
| LD A,(DE) | - - - - - - |
| LD A,I | R R 0 0 R - |
| LD A,(nn) | - - - - - - |
| LD A,R | R R 0 0 R - |
| LD(BC),A | - - - - - - |
| LD(DE),A | - - - - - - |
| LD(HL),n | - - - - - - |
| LD dd,nn | - - - - - - |
| LD HL,(nn) | - - - - - - |
| LD (HL),r | - - - - - - |
| LD I,A | - - - - - - |
| LD IX,nn | - - - - - - |
| LD IX,(nn) | - - - - - - |
| LD (IX+d),n | - - - - - - |
| LD (IX+d),r | - - - - - - |
| LD IY,nn | - - - - - - |
| LD IY,(nn) | - - - - - - |
| LD(IY+d),n | - - - - - - |
| LD(IY+d),r | - - - - - - |
| LD(nn),A | - - - - - - |
| LD(nn),dd | - - - - - - |
| LD(nn),HL | - - - - - - |
| LD(nn),IX | - - - - - - |
| LD(nn),IY | - - - - - - |
| LD R,A | - - - - - - |
| LD r,(HL) | - - - - - - |
| LD r,(IX+d) | - - - - - - |
| LD r,(IY+d) | - - - - - - |
| LD r,n | - - - - - - |
| LD r,r' | - - - - - - |
| LD SP,HL | - - - - - - |
| LD SP,IX | - - - - - - |
| LD SP,IY | - - - - - - |
| LDD | - - 0 0 R - |
| LDDR | - - 0 0 R - |
| LDI | - - 0 0 R - |
| LDIR | - - 0 0 R - |
| NEG | R R R 1 R R |
| NOP | - - - - - - |
| OR s | R R 0 0 R 0 |
| OTDR | - 1 - 1 - - |
| OTIR | - 1 - 1 - - |
| OUT (C),r | - - - - - - |
| OUT (n),A | - - - - - - |
| OUTD | - R - 1 - - |
| OUTI | - R - 1 - - |
| POP IX | - - - - - - |
| POP IY | - - - - - - |
| POP qq | - - - - - - |
| PUSH IX | - - - - - - |
| PUSH IY | - - - - - - |
| PUSH qq | - - - - - - |
| RES b,m | - - - - - - |
| RET | - - - - - - |
| RET cc | - - - - - - |
| RETI | - - - - - - |
| RETN | - - - - - - |
| RL m | R R 0 0 R R |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RLA | - - 0 0 - R | | SET b,(HL) | - - - - - - |
| RLC (HL) | R R 0 0 R R | | SET b,(IY+d) | - - - - - - |
| RLC (IX+d) | R R 0 0 R R | | SET b,r | - - - - - - |
| RLC (IY+d) | R R 0 0 R R | | SLA m | R R 0 0 R R |
| RLC r | R R 0 0 R R | | SRA m | R R 0 0 R R |
| RLCA | - - 0 0 - R | | SRL m | R R 0 0 R R |
| RLD | R R 0 0 R R | | SUB s | R R R 1 R R |
| RR m | R R 0 0 R R | | XOR s | R R R 1 R R |
| RRA | - - 0 0 - R | | SET b,(IX+d) | - - - - - - |
| RRC m | R R 0 0 R R | | | |
| RRCA | - - 0 0 - R | | | |
| RRD | R R 0 0 R R | | | |
| RST p | - - - - - - | | | |
| SBC A,s | R R R 1 R R | | | |
| SBC HL,ss | R R R 1 R R | | | |
| SCF | - - 0 0 - 1 | | | |

ABBREVIAZIONI:

| | |
|---|---|
| r | Registro: A, B, C, D, E, H o L. |
| n | Valore nell'intervallo 0 .. 255 (senza segno). |
| s | r,n,(HL),(IX+d) o (IY+d). |
| m | A, B, C, E, H, L, (HL), (IX+d), (IY+d). |
| dd | BC, DE, HL, SP [LD HL,(dd) LD(nn),dd]. |
| nn | Valore nell'intervallo 0 .. 65535. |
| qq | BC, DE, HL, AF (POP qq, PUSH qq). |
| ss | BC, DE, HL, SP (ADC HL,ss; ADD HL,ss; DEC ss; INC ss; SBC HL,ss). |
| pp | BC, DE, IX, SP (ADD IX,pp). |
| rr | BC, DE, IY, SP (ADD IY,rr). |
| b | Bit da 0 a 7 |
| e | Uno spiazzamento intero compreso nell'intervallo —126 .. 129. |
| p | Un indirizzo di pagina zero dato nell'istruzione RESTART. Deve essere compreso tra 0 e 56 e deve essere multiplo di otto. |
| cc | Codice di condizione: C, NC, Z, NZ, P, M, PE, PO. |

# APPENDICE E

## Estratto dal Manuale del TMS 9118/9128/9128 Data

Riprodotto per gentile concessione della Texas Instruments Ltd.

2. Architettura

Il Processore Video (Video Display Processor = VDP) TMS9118 è progettato per fornire una semplice interfaccia tra un microprocessore e un apparecchio televisivo o un monitor a colori. Viceversa i TMS9128/9129 VDP sono progettati per essere una semplice interfaccia tra un microprocessore e un monitor RGB o un codificatore video che produca il segnale video per guidare il monitor stesso. La Figura 2.1 è un diagramma a blocchi delle parti principali dell'architettura del VDP d'interfaccia tra CPU, VRAM e televisore a colori.

2.1 Interfaccia con la CPU

Il VDP s'interfaccia alla CPU usando un bus dati su 8 bit, bidirezionale, tre linee di controllo e un'interruzione. Usando queste interfacce la CPU è in grado di effettuare quattro tipi di operazioni:
1) Scrivere in uno degli otto registri a sola scrittura del VDP.
2) Leggere il registro di stato del VDP.
3) Scrivere nella VRAM dati (byte) da visualizzare.
4) Leggere dati (byte) dalla VRAM.
Ciascuna di queste operazioni implica che vengano effettuati uno o più trasferimenti di dati sul bus d'interfaccia. L'interpretazione del dato trasferito è determinata dallo stato delle tre linee di controllo del VDP.

Nota

La CPU può comunicare con il VDP sia simultaneamente che in modo asincrono rispetto alle operazioni di refresh dello schermo compiute dal VDP. Il VDP effettua le gestione della memoria e consente alla CPU di accedere alla VRAM ad intervalli periodici, anche a metà di una scansione video.

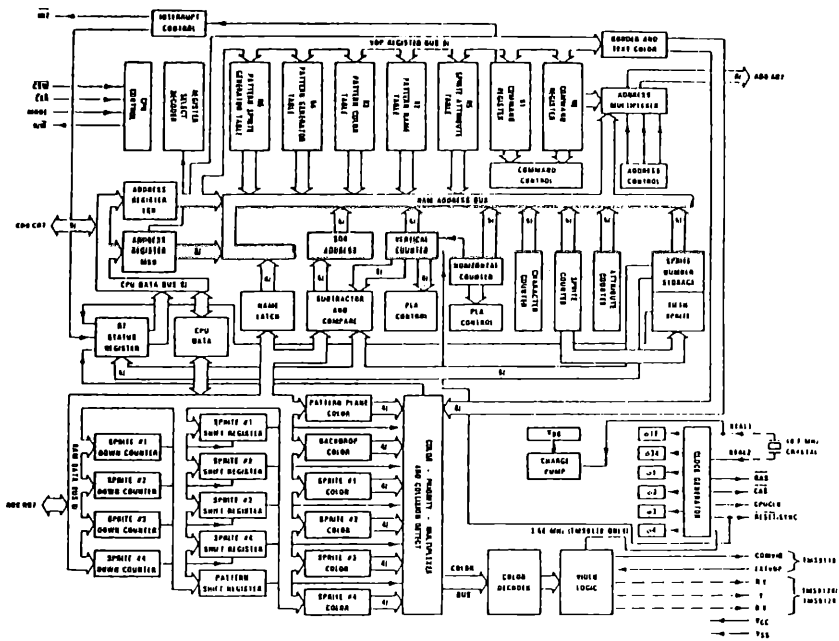



*Fig. 2-1 - Diagramma a blocchi del VDP.*

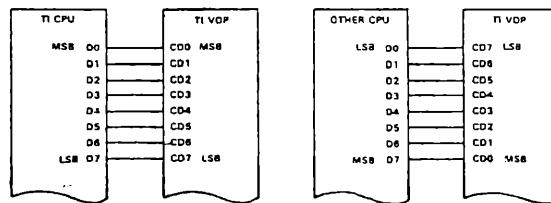



*Fig. 2-2 - Connessione tra CPU e VDP.*

### 2.1.1 Il bus dati tra CPU e VDP

La CPU trasferisce dati tra sé stessa e il VDP attraverso un bus dati bidirezionale su 8 bit. Come viene mostrato nelle tabelle 2-1 e 2-2 gli otto bit del bus vengono denotati da CD0 (MSB) a CD7 (LSB).

Nota

In questo manuale l'interfaccia tra CPU e VDP viene descritta assumendo che vengano utilizzati integrati TI. Altri costruttori di CPU devono assegnare le linee dati D0 al LSB e D7 al MSB. Il modo corretto viene mostrato nella Figura 2-2. In caso di connessione scorretta non si ottiene alcuna visualizzazione sullo schermo.

### 2.1.2 Segnale di controllo d'interfaccia con la CPU

Gli ingressi CSW, CSR e MODE controllano il tipo e la direzione del trasferimento dati. La linea CSW seleziona un'operazione di scrittura dalla CPU al VDP. Quando è attiva (bassa) gli otto bit C0 - C7 vengono sentiti dal VDP. La linea CSR seleziona un'operazione di scrittura da parte della CPU nel VDP. Quando è attiva (bassa), il VDP emette gli otto bit C0-C7 verso la CPU. Le linee CSW e CSR non possono mai essere contemporaneamente attive. Se ciò accade i dati trasferiti non sono validi.
La linea MODE determina la sorgente o la destinazione di un trasferimento dati in scrittura o in lettura. Di solito questo ingresso è legato ad una linea di indirizzo di basso ordine della CPU.

### 2.1.3 Registri del VDP

Il VDP è dotato di otto registri a sola scrittura e un registro di stato a sola lettura. I registri a sola scrittura controllano l'operazione effettuata dal VDP e determinano l'allocazione della VRAM. Il registro di stato memorizza la condizione d'interruzione pendente, la collisione di sprite e la condizione di quinto sprite su una medesima linea di schermo. Le sezioni 2.2 e 2.3 contengono rispettivamente descrizioni più dettagliate dei registri a sola scrittura e del registro di stato.
Un valore può essere caricato in uno qualsiasi degli otto registri a sola scrittura usando un doppio trasferimento dati (di 8 bit) dalla CPU. Nella tabella 2-3 viene descritto il formato richiesto per ciascuno di questi due byte: il primo è il dato vero e proprio, il secondo indica la sua destinazione. Il MSB del secondo byte deve essere a 1.

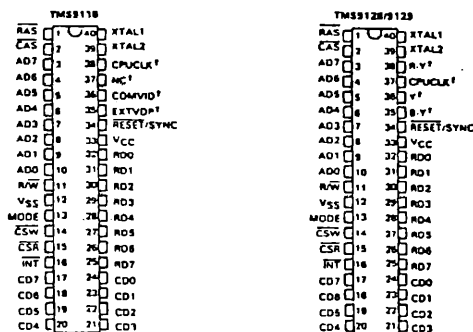



*Fig. 2-3 - Assegnazione dei piedini nel TMS 9118/9128/9129.*

I successivi quattro bit sono a 0, mentre gli ultimi tre contengono il numero di registro (da 0 a 7). La linea MODE è alta per il trasferimento di entrambi i dati.

Nota

Se i due bit più significativi del secondo byte indicano che è in corso una scrittura di un registro, il primo byte non è interpretato come byte d'indirizzo della CPU.

### 2.1.4 La CPU scrive nella VRAM

La CPU trasferisce dati nella VRAM attraverso il VDP usando un registro di 14 bit autoincrementante. L'impostazione del registro d'indirizzo richiede un trasferimento di due byte. Ciascuno di questi richiede un tempo di due microsecondi, per un totale di quattro microsecondi (si veda la tabella 2-3). Un ulteriore trasferimento di un byte è necessario per scrivere effettivamente il dato nel byte della VRAM indirizzato. Il tempo necessario dipende dallo stato della linea MODE. Quindi il registro d'indirizzo si autoincrementa. Successivi trasferimenti sequenziali richiedono solo il trasferimento di un byte, poiché l'indirizzo è già correttamente impostato. Durante la fase di preparazione dell'indirizzo, i due bit più significativi del secondo byte d'indirizzo devono essere 0 e 1, rispettivamente. La linea MODE è alta per entrambi i trasferimenti che concernono l'indirizzo, mentre è bassa per il trasferimento del dato. La CSW è usata in tutti i trasferimenti per scrivere i dati su 8 bit nel VDP (si veda la tabella 2-3).

*Tab. 2-3 - Sequenze di trasferimento dati e di "setup" tra CPU e VDP*

| OPERAZIONE | BIT 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | CSW | CSR | MODO | REG. TIME |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Scrittura in registri VDP | | | | | | | | | | | | |
| Byte 1: Dato da scrivere | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | 0 | 1 | 1 | 2 $\mu s$ |
| Byte 2: Selezione del registro | 1 | 0 | 0 | 0 | 0 | $RS_0$ | $RS_1$ | $RS_2$ | 0 | 1 | 1 | 2 $\mu s$ |
| Scrittura nella VRAM | | | | | | | | | | | | |
| Byte 1: Impostazione dell'indirizzo | $A_6$ | $A_7$ | $A_8$ | $A_9$ | $A_{10}$ | $A_{11}$ | $A_{12}$ | $A_{13}$ | 0 | 1 | 1 | 2 $\mu s$ |
| Byte 2: Impostazione dell'indirizzo | 0 | 1 | $A_0$ | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | 0 | 1 | 1 | 2 $\mu s$ |
| Byte 3: Dato da scrivere | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | 0 | 1 | 0 | — |
| Lettura registro di stato | | | | | | | | | | | | |
| Byte 1: Dato da leggere | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | 1 | 0 | 1 | 2 $\mu s$ |
| Lettura della VRAM | | | | | | | | | | | | |
| Byte 1: Impostazione dell'indirizzo | $A_6$ | $A_7$ | $A_8$ | $A_9$ | $A_{10}$ | $A_{11}$ | $A_{12}$ | $A_{13}$ | 0 | 1 | 1 | 2 $\mu s$ |
| Byte 2: Impostazione dell'indirizzo | 0 | 0 | $A_0$ | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | 0 | 1 | 1 | — |
| Byte 3: Dato da leggere | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | 1 | 0 | 0 | — |

### 2.1.5 La CPU legge il registro di stato del VDP

Come viene illustrato nella tabella 2-3, la CPU può leggere il registro di stato del VDP con un unico trasferimento di dati. La linea MODE è alta. La linea CSR viene usata per segnalare al VDP che è richiesta un'operazione di lettura. Ciascuna lettura del registro di stato richiede un minimo di due microsecondi.

### 2.1.6 La CPU legge dalla VRAM

La CPU legge dalla VRAM attraverso il VDP usando il registro d'indirizzo ad autoincremento. Una volta che l'indirizzo è stato impostato, il trasferimento del byte di dato indirizzato nella VRAM è tutto ciò che è necessario fare. Quindi il registro d'indirizzo si autoincrementa. Successivi trasferimenti sequenziali per leggere dati nella VRAM richiedono quindi il trasferimento di un solo byte poiché l'indirizzo è già predisposto. Durante la preparazione del registro d'indirizzo i due bit più significativi del secondo byte di indirizzo devono essere posti a 0. Impostando così il registro d'indirizzo s'inizia un ciclo di lettura dalla VRAM, e il dato letto è disponibile per il primo trasferimento dati verso la CPU (si veda la tabella

2-3). La linea MODE è alta per il trasferimento dei byte d'indirizzo e bassa per il trasferimento del dato.
La CPU interagisce con la memoria VRAM attraverso il VDP. Occorrono quattro microsecondi per la preparazione dell'indirizzo per una scrittura e due microsecondi per la preparazione dell'indirizzo per una lettura. Il tempo necessario alla CPU per trasferire un byte di dato da o verso la memoria VRAM può variare da uno a otto microsecondi. Una volta che al VDP è stato "detto" di leggere o scrivere un dato dalla VRAM, ci vogliono circa due microsecondi prima che il VDP sia pronto per il trasferimento effettivo del dato. Questo tempo è misurato a partire dalla salita iniziale della linea CSW sul byte 3 per una scrittura e dalla salita iniziale della linea CSR sul byte 2 per una lettura. Oltre a questo ritardo di due microsecondi, il VDP deve attendere una "finestra d'accesso" per la CPU (cioè un periodo di tempo in cui il VDP stesso non è impegnato nel refresh della memoria o nella visualizzazione su video) perché il dato possa essere letto o scritto.
Il caso peggiore tra la presentazione di due successive finestre di accesso si ha nella modalità Grafica I e II quando si stanno usando gli sprite (si veda la tabella 2-4). Durante la visualizzazione attiva, le finestre per la CPU si presentano ogni 16 cicli di memoria, portando ad un ritardo massimo di 6 microsecondi (un ciclo di memoria dura circa 372 nanosecondi). Nella modalità testo la finestra di accesso per la CPU si presenta almeno ogni tre cicli di memoria, con un ritardo pari a 1.1 microsecondo nel caso peggiore. Infine nella modalità multicolore la finestra per la CPU si presenta almeno una volta ogni quattro cicli di memoria.
Se l'utente ha necessità di accedere alla memoria in due microsecondi, ci

*Tab. 2-4 - Tabella dei tempi di ritardo nell'accesso alla memoria*

| Condizione | Modalità | Ritardo VDP | Tempo d'attesa per finestra d'accesso | Tempo totale |
|---|---|---|---|---|
| Area attiva di visualizzazione | testo | 2 µs | 0-1.1 µs | 2-3.1 µs |
| Area attiva di visualizzazione | Grafica I, II | 2 µs | 0-5.95 µs | 2-8 µs |
| 4.300 µs dopo segnale d'interruzione | tutte | 2 µs | 0 µs | 2 µs |
| Blank bit = 0 Registro 1 | tutte | 2 µs | 0 µs | 2 µs |
| Area attiva di visualizzazione | multicolore | 2 µs | 0-1.5 µs | 2-3.5 µs |

sono due situazioni in cui il tempo d'attesa per l'accesso è effettivamente nullo. Entrambe non dipendono dalla modalità corrente di visualizzazione. La prima si presenta quando il bit "blank" del registro 1 è 0. In questa situazione sullo schermo viene visualizzato solo il colore del bordo, e il VDP non deve attendere per una finestra d'accesso della CPU.
La seconda si presenta durante l'intervallo di "ritorno" ("flyback") verticale della scansione del video. Il VDP instaura in questo caso un'interruzione al termine di ogni area attiva. Questo segnale indica che il VDP sta entrando nella fase di ritorno verticale, e che per i prossimi 4.3 millisecondi non vi è tempo di attesa per una finestra d'accesso. Se l'utente vuole che la CPU acceda alla memoria in questo intervallo, la CPU di controllo deve accorgersi del segnale d'interruzione lanciato dal VDP (può controllarlo periodicamente, in "polling", o considerarlo come un'interruzione in ingresso).
Il programma che si accorge dell'interruzione deve tenere presente il suo proprio ritardo nel rispondere all'interruzione stessa e tener presente quanto tempo rimane dei 4.3 millisecondi del periodo di refresh di cui si diceva prima. La CPU deve scrivere 1 nel bit di abilitazione delle interruzioni del registro 1 in fase d'inizializzazione per consentire l'interruzione ad ogni scansione video. Deve poi leggere il registro di stato ad ogni interruzione.

### 2.1.7 CPU guidate da interruzioni

In un ambiente guidato da interruzioni (cioè nel quale la CPU accetta le interruzioni esterne) è possibile che una interruzione si verifichi prima che una delle sequenze mostrate nella tabella 2-3 finisca. Per esempio, può capitare una interruzione immediatamente dopo il caricamento del byte 1 e 2 d'indirizzo in una operazione di scrittura su VRAM. In questo caso la routine di gestione dell'interruzione non sa a quale punto della sequenza ci si trovava. Si rende quindi necessario disabilitare le interruzioni, per poi riabilitarle, durante ogni sequenza di preparazione. In tal modo si previene una perdita di continuità tra la CPU e il VDP.
Se questa continuità non è importante, si può leggere il registro di stato. La logica interna dell'interfaccia della CPU viene in questo modo rinizializzata e accetta il prossimo byte come il primo evento di una nuova sessione d'interfaccia tra CPU e VDP. Se nessuna delle due tecniche citate è accettabile, è necessario usare l'interrogazione periodica ("polling").

### 2.1.8 Interruzione del VDP

Il pin di output INT del VDP viene usato per generare un'interruzione al

termine di ogni scansione della zona attiva (approssimativamente ogni 1/60 di secondo per TMS 9118/9128 e 1/50 di secondo per TMS 9129). Il pin INT è attivo se è stato posto a 1 il bit di abilitazione delle interruzioni (EI) del registro 1 del VDP e se il flag F del registro di stato è anch'esso posto a 1. Le interruzioni sono rimosse quando si effettua una lettura del registro di stato.
L'interruzione del VDP si presenta alla fine della visualizzazione del piano dei caratteri attivo, prima che vengano visualizzate le ultime linee di sfondo. Questa interruzione può essere usata per muovere gli sprite o aggiornare la visualizzazione sul piano dei modelli, ma non è utile per cambiare rapidamente il colore dello sfondo.

### 2.1.9 RESET/SYNC

Il VDP viene inizializzato esternamente quando viene tenuto attivo (basso) per un minimo di tre microsecondi l'ingresso RESET/SYNC. Il RESET esterno sincronizza tutti i temporizzatori, imposta i contatori verticali e orizzontali a valori noti, e pulisce i registri 0 e 1 del VDP. Lo schermo viene automaticamente pulito poiché il bit "blank" del registro 1 viene posto a 0. Comunque il VDP continua il refresh della memoria anche se lo schermo viene pulito. Per ripristinare la schermata è sufficiente scrivere gli opportuni valori nei registri 0 e 1. Mentre la linea RESET/SYNC è attiva, il VDP non fa il refresh della memoria.

## 2.2 Registri a sola scrittura

Gli otto registri a sola scrittura del VDP sono mostrati nella Figura 2-4. Il loro valore viene impostato dalla CPU secondo le modalità descritte nella sezione 2.1.3. I registri 0 e 1 contengono dei flag per abilitare o disabilitare varie funzioni e modalità operazionali del VDP. I registri dal 2 al 6 definiscono gli indirizzi di partenza per numerosi sottoblocchi logici della VRAM. Questi sottoblocchi costituiscono le tabelle che vengono usate per produrre l'immagine desiderata sullo schermo. Il contenuto di queste tabella è determinato dal microprocessore. Il registro 7 viene usato per definire i colori del testo e dello sfondo.
I registri vengono descritti singolarmente nelle sezioni che seguono:

### 2.2.1 Registro 0

Il registro 0 contiene due bit di controllo delle opzioni del VDP. Tutti gli

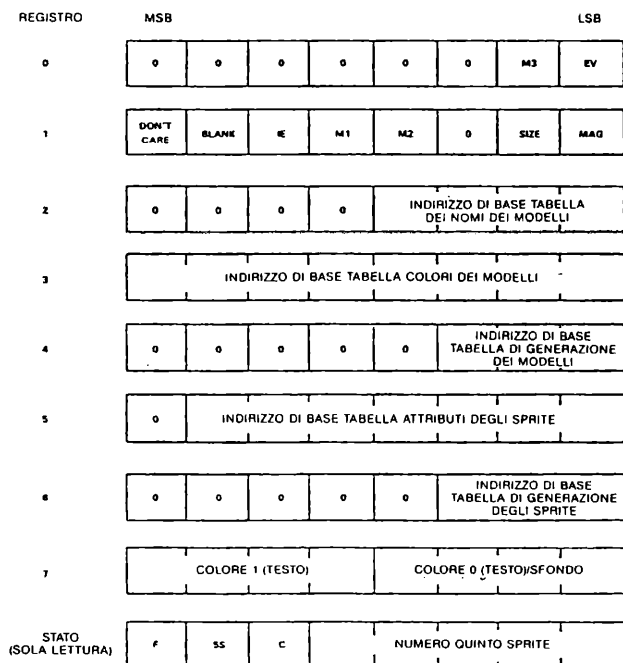



*Fig. 2-4 - Registri del VDP.*

altri bit sono riservati per espansioni future e devono essere posti a zero.
BIT 6 - M3 (bit 3 di modalità) (si veda sezione 2.2.2 per la descrizione).
BIT 7 - Abilitazione/disabilitazione di un VDP esterno (si veda sezione 3.7 per la descrizione). 0 disabilita il VDP esterno, 1 lo abilita.

2.2.2 Registro 1 (contiene 6 bit di controllo di opzioni del VDP)

BIT 0 - Non usato.
BIT 1 - Abilitazione/disabilitazione BLANK. Se posto a 0 fa in modo che venga pulita l'area attiva di visualizzazione, lasciando sullo schermo solo il colore del bordo. Posto a 1 abilita la visualizzazione nell'area attiva.
BIT 2 - EI (Enable Interrupt = Abilita le interruzioni): posto a zero disabilita le interruzioni del VDP, viceversa se posto a 1.
BIT 3, 4 - M1 e M2 (bit 1 e 2 di modalità): i bit M1, M2 e M3 determinano il modo operativo del VDP:

| M1 | M2 | M3 | |
|---|---|---|---|
| 0 | 0 | 0 | Modalità Grafica I |

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | Modalità Grafica II |
| 0 | 1 | 0 | Modalità multicolore |
| 1 | 0 | 0 | Modalità testo |

BIT 5 - Riservato.
BIT 6 - Selettore delle dimensioni degli sprite: posto a 0 fa in modo che siano usati sprite di 8 * 8 pixel; posto a 1 sprite di 16 * 16.
BIT 7 - MAG (ingrandimento degli sprite): posto a 1 raddoppia le dimensioni degli sprite e viceversa le riporta a quelle normali.

### 2.2.3 Registro 2

Il registro 2 definisce l'indirizzo di partenza per il sottoblocco della tabella dei nomi dei modelli. La tabella dei nomi dei modelli contiene il nome o il puntatore alla definizione del modello contenuta nella tabella di generazione dei modelli. Il valore contenuto nel registro 2 è compreso nell'intervallo da 0 a 15. Il contenuto del registro forma i quattro bit più significativi degli indirizzi su 14 bit della tabella dei nomi dei modelli: ne consegue che l'indirizzo di questa tabella è uguale a (Registro 2)* 400 (esadecimale). La tabella 2-5 mostra i possibili indirizzi di partenza del sotto-blocco della tabella dei nomi dei modelli.

Indirizzo di partenza = registro 2 * 400 (esad.)

| R2 | Indirizzo | |
|---|---|---|
| 00 | 0000 | |
| 01 | 0400 | |
| 02 | 0800 | |
| 03 | 0C00 | - Massimo per RAM di 4K |
| 04 | 1000 | |
| 05 | 1400 | |
| 06 | 1800 | |
| 07 | 1C00 | |
| 08 | 2000 | |
| 09 | 2400 | |
| 0A | 2800 | |
| 0B | 2C00 | |
| 0C | 3000 | |
| OD | 3400 | |
| OE | 3800 | |
| OF | 3C00 | - Massimo consentito |

*Tab. 2-5 - Indirizzamento con il registro 2*

### 2.2.4 Registro 3

Il Registro 3 definisce l'indirizzo di partenza del sottoblocco della tabella dei colori dei modelli. Questa tabella definisce quale colore è associato agli 1 e agli 0 della definizione del modello del carattere. Il valore contenuto nel registro 3 è compreso nell'intervallo tra 0 e 255. Questo contenuto costituisce gli otto bit più significativi dell'indirizzo su 14 bit della tabella dei colori dei modelli: quindi il suo indirizzo può essere calcolato come Registro 3 * 40 (esadecimale). La tabella 2-6 mostra i possibili indirizzi di partenza per la tabella dei colori dei modelli, con l'eccezione della modalità Grafica II.

Nota

Il Registro 3 funziona in modo differente nella modalità Grafica II. In questa modalità la tabella dei colori dei modelli può essere allocata solamente a uno tra due indirizzi della VRAM: 0000 e 2000. Se il suo indirizzo è 0000 allora il bit più significativo del registro 3 deve essere posto a 0. Se l'indirizzo è 2000, vale viceversa (lo stesso bit è cioè posto a 1). I bit da 1 a 7 del registro 3 devono essere posti a 1: ne consegue che nella modalità Grafica II gli unici valori del registro 3 che funzionano correttamente sono 7F e FF.

### 2.2.5 Registro 4

Il registro 4 definisce l'indirizzo di partenza per il sottoblocco della tabella di generazione dei modelli. Questa tabella contiene una libreria di modelli (definizioni delle forme dei caratteri) che possono essere visualizzati sullo schermo. Il valore contenuto nel registro 4 è compreso tra 0 e 7. Tale valore costituisce i tre bit più significativi degli indirizzi su 14 bit della tabella di generazione dei modelli: quindi l'indirizzo di partenza di quest'ultima è pari a Registro 4 * 800 (esadecimale). La tabella 2-7 mostra i possibili indirizzi di partenza per il sottoblocco della tabella di generazione dei modelli, con l'eccezione della modalità Grafica II.

Nota

Il registro 4 funziona diversamente quando il VDP è in modalità Grafica II. In questa modalità la tabella di generazione dei modelli può essere solamente posta a due locazioni della VRAM: 0000 e 2000. Se vale il primo caso il bit 5 del registro 4 deve essere 0; se vale il secondo caso (indirizzo 2000) vale viceversa, cioè il bit 5 è posto a 1. In entrambi i casi i bit 6 e

7 devono essere a 1. Quindi nella modalità Grafica II i soli valori accettabili per il registro 4 sono >03 e >07.

Attenzione

Il bit 5 del registro 4 DEVE avere il valore opposto a quello del bit 0 del registro 3, in modo che le due aree di 8K siano separate. Altrimenti la modalità Grafica II non funziona correttamente.

Indirizzo di partenza = Registro 4 * 800 (esadec.)

| R4 | Indirizzo di partenza | |
|---|---|---|
| 00 | 0000 | |
| 01 | 0800 | - Massimo per RAM di 4K |
| 02 | 1000 | |
| 03 | 1800 | |
| 04 | 2000 | |
| 05 | 2800 | |
| 06 | 3000 | |
| 07 | 3800 | - Massimo consentito |

*Tab. 2-7 - Indirizzamento con il registro 4*

## 2.2.6 Registro 5

Il Registro 5 definisce l'indirizzo di partenza del sotto-blocco della tabella degli attributi degli sprite. Questa tabella specifica dove i vari sprite debbano essere visualizzati sullo schermo. Il valore contenuto nel registro è compreso tra 0 e 127. Esso forma i sette bit più significativi degli indirizzi su 14 bit della tabella degli attributi degli sprite: quindi l'indirizzo di partenza è uguale a registro 5 * 80 (esadecimale).

## 2.2.7 Registro 6

Il registro 6 definisce l'indirizzo di partenza del sotto-blocco della tabella di generazione dei modelli degli sprite. Questa tabella contiene la descrizione della forma degli sprite. Il valore contenuto nel registro è compreso tra 0 e 7. Esso forma i tre bit più significativi degli indirizzi su 14 bit della tabella di generazione dei modelli degli sprite: quindi l'indirizzo di partenza è uguale a registro 6 * 800 (esadecimale). La tabella 2-9 mostra i possibili indirizzi di partenza per la suddetta tabella:

*Tab. 2-9 - Indirizzamento con il registro 6*

Indirizzo di partenza = Registro 6 * 800 (esadec.)

| R4 | Indirizzo di partenza | |
|---|---|---|
| 00 | 0000 | |
| 01 | 0800 | - Massimo per RAM di 4K |
| 02 | 1000 | |
| 03 | 1800 | |
| 04 | 2000 | |
| 05 | 2800 | |
| 06 | 3000 | |
| 07 | 3800 | - Massimo consentito |

### 2.2.8 Registro 7

I 4 bit di ordine più alto del registro 7 contengono il codice del colore 1 nella modalità testo. Gli altri 4 contengono il codice per il colore 0 nella modalità testo, il colore del bordo e dello sfondo nelle altre modalità.

## 2.3 REGISTRO DI STATO

Il VDP ha un solo registro di stato di 8-bit, a sola lettura che può essere acceduto dalla CPU. Il registro di stato contiene il flag di interruzione (F), il flag di collisione di sprite (C), il flag di quinto sprite (5S), e il numero del quinto sprite (se esiste). Il formato del registro di stato è mostrato di seguito e viene discusso nei paragrafi che seguono.

| F | 5S | C | Numero quinto sprite |
|---|---|---|---|

In qualsiasi momento il registro di può essere letto per controllare i flag F,C e 5S. La lettura del registro implica che venga azzerato il flag di interruzione F.

Attenzione

Letture asincrone causano l'azzeramento del flag F, che viene quindi perso. Quindi il registro di stato deve essere letto solo quando il VDP ha una interruzione pendente.

### 2.3.1 Il flag d'interruzione (F)

Il flag F d'interruzione viene posto a 1 alla fine di ogni scansione dell'ultima linea dell'area attiva del video. Viene reimpostato a 0 dopo che il registro di stato è stato letto o quando il VDP viene reinizializzato esternamente. Se nel registro 1 del VDP è attivo (cioè è posto a 1) il bit abilitazione d'interruzione, la linea di uscita di interruzione del VDP (INT) viene tenuta attiva (bassa) fintantochè il flag F d'interruzione è posto a 1.

NOTA

È necessario leggere il registro di stato ad ogni scansione video per ripristinare lo stato delle interruzioni e poter ricevere le successive.

### 2.3.2 Il flag di collisione di sprite (C)

Il flag C di collisione di sprite del registro di stato viene posto a 1 quando vi è sovrapposizione di due o più sprite. Questa situazione si presenta quando due sprite sullo schermo hanno almeno un pixel coincidente. Vengono considerati anche gli sprite definiti di colore trasparente, così come quelli che sono completamente o parzialmente fuori dallo schermo. Per rendere invisibile uno sprite è necessario definirlo con colore trasparente. Il flag C viene rimesso a 0 quando il registro di stato viene letto o quando il VDP viene reinizializzato esternamente. Porre a 1 il flag C non causa un'interruzione. Gli sprite che si trovano oltre il termine (D0) della tabella degli attributi degli sprite non vengono considerati. Se, per esempio, un terminatore D0 viene messo nel byte di posizione verticale dello sprite 3 nella tabella degli attributi degli sprite, quelli di numero da 4 a 31 non vengono visualizzati.

Attenzione

Subito dopo l'accensione del sistema, il registro di stato dovrebbe essere letto per assicurarsi che sia stato posto a 0 il flag di collisione.
Il VDP controlla le eventuali collisioni in ogni possibile pixel durante la generazione del pixel stesso, indipendetemente dalla sua posizione sullo schermo. Questo accade ogni 1/60 di secondo per i TMS9118 e TMS9128, e 1/50 di secondo per il TMS 9129. Quindi quando durante questi intervalli si muove uno sprite per più di una posizione, è possibile che si verifichi la sovrapposizione di più pixel o addirittura che uno sprite sia passato completamente sopra un altro prima che il VDP controlli eventuali collisioni.

### 2.3.3 Numero e flag (5S) del quinto sprite

Il flag di quinto sprite (5S) del registro di stato viene posto a 1 quando ci

sono cinque o più sprite sulla medesima linea orizzontale (linee da 1 a 192) e il flag di interruzione è posto a zero. Il flag di quinto sprite viene impostato a 1 anche se gli sprites sono posizionati fuori dello schermo. Il flag 5S viene reimpostato a zero quando viene letto il registro di stato o il VDP viene rinizializzato dall'esterno. Il numero del quinto sprite è posto nei 5 bit meno significativi del registro di stato, ovviamente quando è a 1 il flag di quinto byte. Porre a 1 il flag di quinto sprite non provoca un'interruzione.

### 2.3.4 Oscillatore e temporizzatore

Il VDP è stato progettato per funzionare con un temporizzatore da 10.738635 (+/— 0.0005) MHz per generare i segnali interni di sincronismo richiesti. Un cristallo a frequenza-fondamentale e modo-parallelo è necessario per l'oscillatore interno, che genera la sincronizzazione per tutte le operazioni di sistema. Questa temporizzazione principale viene divisa per due per generare la tamporizzazione dei pixel (5.3 MHz) e per tre per fornire il segnale CPUCLK (3.58 MHz solo per TMS 9118).

# APPENDICE F

## Estratto dal Manuale del generatore programmabile di suoni AY-3-8910

(riprodotto per gentile concessione della General Instruments Microelectronics Ltd.)

### 1.2 Caratteristiche

— Completo controllo software della generazione del suono.
— Interfaccia con la maggior parte dei microprocessori a 8 e 16 bit.
— Tre uscite analogiche indipendenti.
— Due porte di I/O ad utilizzo generale su 8 bit (AY-3-8910).
— Una porta di I/O ad utilizzo generale su 8 bit (AY-3-8912).
— Alimentazione +5 Volt unica.

### 1.3 Scopo del manuale

Questo manuale si prefigge d'illustrare in che modo programmare il generatore programmabile di suoni AY-3-8910/8912 per ottenere i risultati sonori desiderati. Tutti i programmi e i progetti hardware sono stati provati allo scopo di verificarne l'applicabilità pratica, anziché solo teorica. Sebbene le tecniche presentate consentano di ottenere ottimi risultati, le possibilità di generazioni di suoni sono talmente vaste che è corretto considerare questo manuale solo come un'introduzione alle potenzialità applicative del PSG.

### 2.2 Assegnazioni dei piedini

L'AY-3-8910 è un integrato che viene fornito in un package dual-in-line a 40 piedini che vengono logicamente assegnati come illustrato nella figura

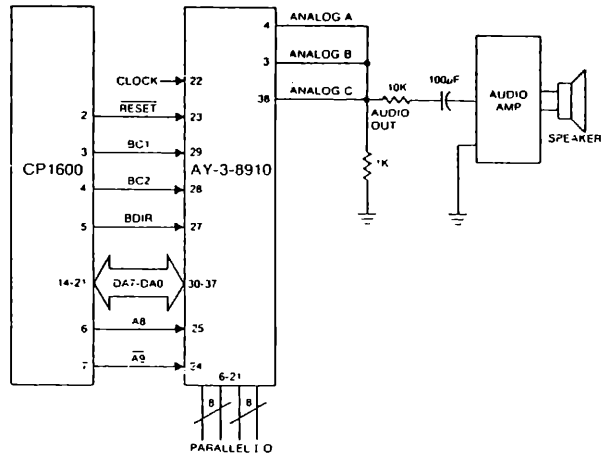



*Fig. 1 - Diagramma di sistema tipico.*

4. Viceversa l'AY-3-8912 è un package dual-in-line a 28 piedini logicamente assegnati come illustrato nella figura 5.

## 2.3 Funzioni dei piedini

DA7-DA0 (ingresso/uscita/alta impedenza): piedini 30-37 (AY-3-8910)
Dato/Indirizzo 7--0: piedini 21-28 (AY-3-8912)

Queste otto linee costituiscono il bus di 8 bit bidirezionale usato dal microprocessore per inviare al PSG sia dati che indirizzi e per ricevere dati dal PSG stesso. Nella modalità dati, DA7--DA0 corrispondono ai bit B7--B0 del vettore dei registri. Nella modalità indirizzo DA3--DA0 selezionano il numero di registro (0--17 ottale) e DA7--DA4 in unione con gli ingressi di indirizzo A9 e A8 formano l'indirizzo d'ordine alto (selezione dell'integrato).

A8 (ingresso): piedino 25 (AY-3-8910)
piedino 17 (AY-3-8912)
A9 (ingresso): piedino 24 (AY-3-8910)
non disponibile sull'AY-3-8912

Indirizzo 9, Indirizzo 8
Questi bit "extra" d'indirizzo sono stati messi a disposizione per poter posizionare il PSG (occupando uno spazio di memoria di 16 parole) all'interno di uno spazio di memoria totale di 1024 parole piuttosto che in uno spazio di sole 256 parole, così come consentito dal solo utilizzo dei bit di indiriz-

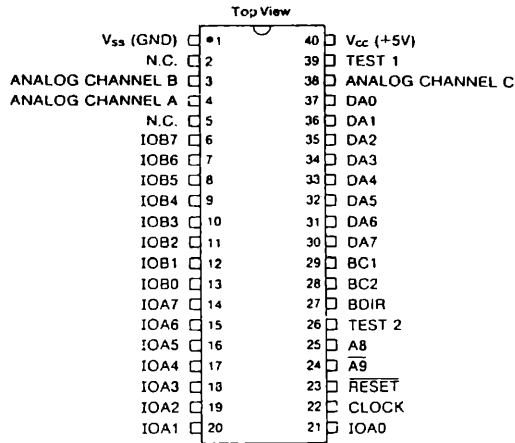



*Fig. 4 - AY-3-8910 assegnazioni dei piedini.*

Top View

| | | | |
|---|---|---|---|
| ANALOG CHANNEL C | 1 | 28 | DA0 |
| TEST 1 | 2 | 27 | DA1 |
| $V_{CC}$ (+5V) | 3 | 26 | DA2 |
| ANALOG CHANNEL B | 4 | 25 | DA3 |
| ANALOG CHANNEL A | 5 | 24 | DA4 |
| $V_{SS}$ (GND) | 6 | 23 | DA5 |
| IOA7 | 7 | 22 | DA6 |
| IOA6 | 8 | 21 | DA7 |
| IOA5 | 9 | 20 | BC1 |
| IOA4 | 10 | 19 | BC2 |
| IOA3 | 11 | 18 | BDIR |
| IOA2 | 12 | 17 | A8 |
| IOA1 | 13 | 16 | $\overline{RESET}$ |
| IOA0 | 14 | 15 | CLOCK |

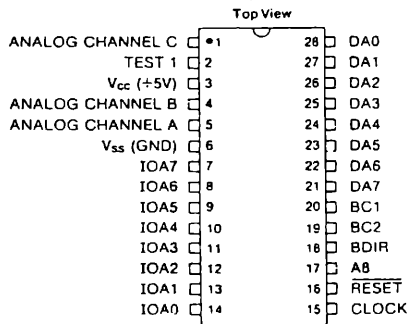



*Fig. 5 - AY-3-8912 assegnazioni dei piedini.*

zo DA7--DA0. Se la dimensione della memoria non richiede l'uso di queste linee d'indirizzo extra, esse possono essere lasciate non connesse poiché entrambe sono fornite di un resistore interno all'integrato di tipo "pull-down" (A9) o "pull-up" (A8). In un ambiente "rumoroso" si raccomanda di connettere sia A9 che A8, se non sono usati, rispettivamente alla presa di a terra e a +5V.

RESET (ingresso): piedino 23 (AY-3-8910)
piedino 16 (AY-3-8912)

Per un'inizializzazione logica come quella che avviene al momento dell'accensione, si deve applicare uno "0" logico (terra) al piedino di RESET: tutti i registri vengono di conseguenza azzerati. Il piedino di RESET è fornito di un resistore interno all'integrato di tipo "pull-up".

CLOCK (ingresso): piedino 22 (AY-3-8910);
piedino 15 (AY-3-8912)

Questo ingresso, compatibile TTL, fornisce la sincronizzazione per i generatori di suono, rumore e forma d'inviluppo.

BDIR, BC2, BC1 (ingressi): piedini 27, 28, 29 (AY-3-8910);
piedini 18, 19, 20 (AY-3-8912)

Bus DIRection, Bus Control 1,2
Questi segnali di controllo del bus vengono generati direttamente da microprocessori della serie GI CP1600 per controllare tutte le operazioni, interne ed esterne, del bus nel PSG. Usando processori di tipo diverso, questi segnali possono essere forniti per mezzo di linee di bus analoghe o simulandoli sulle linee di I/O del processore. Il PSG decodifica questi segnali come qui illustrato:

| *BDIR* | *BC2* | *BC1* | *Funzione CP1600* | *Funzione PSG* |
|---|---|---|---|---|
| 0 | 0 | 0 | NACT | INATTIVO. Si veda 010 (IAB). |
| 0 | 0 | 1 | ADAR | INDIRIZZO ALLACCIAMENTO. Si veda 111(INTAK). |
| 0 | 1 | 0 | IAB | INATTIVO. Il bus CPU/PSG è inattivo. DA7--DA0 sono nello stato ad alta impedenza. |
| 0 | 1 | 1 | DTB | LETTURA DAL PSG. Questo segnala fa sì che il contenuto del registro indirizzato venga messo sul bus PSG/CPU. DA7 -- DA0 sono in modalità d'uscita. |
| 1 | 0 | 0 | BAR | INDIRIZZO ALLACCIAMENTO. Si veda 111(INTAK). |
| 1 | 0 | 1 | DW | INATTIVO. Si veda 010. |
| 1 | 1 | 0 | DWS | SCRITTURA SUL PSG. Questo segnale indica che sul bus c'è un dato che va connesso al registro correntemente indirizzato. DA7--DA0 sono in modalità d'ingresso. |
| 1 | 1 | 1 | INTAK | INDIRIZZO ALLACCIAMENTO. Questo segnale indica che sul bus vi è un indirizzo a cui la CPU deve allacciarsi nel PSG. |

Quando ci s'interfaccia ad un processore differente dal CP 1600, è suffi-

ciente simulare la decodifica sopra riportata; inoltre possono essere sfruttate le ridondanze dei segnali di controllo del bus rispetto alle funzioni del PSG poiché solo quattro tra le possibili funzioni del bus sono utilizzate dal PSG. Da ciò si può ricavare una semplificazione della programmazione del controllo del bus in cui sono richiesti solo la linea BDIR e BC1, ove la linea BC2 è connessa a +5V:

| *BDIR* | *BC2* | *BC1* | *Funzione PSG* |
|---|---|---|---|
| 0 | 1 | 0 | Inattivo |
| 0 | 1 | 1 | Lettura dal PSG |
| 1 | 1 | 0 | Scrittura nel PSG |
| 1 | 1 | 1 | Indirizzo allacciamento |

Canali Analogici A, B, C: (uscite): piedini 4, 3, 38 (AY-3-8910)
piedini 5, 4, 1 (AY-3-8912)

Ciascuno di questi segnali è l'uscita di un corrispondente convertitore analogico/digitale; complessivamente forniscono il segnale che rappresenta la complessa forma d'onda del suono generato dal PSG.

IOA7--IOAO (ingresso/uscita): piedini 14--21 (AY-3-8910)
piedini 7--14 (AY-3-8912)
IOB7--IOB0 (ingresso/uscita): piedini 6--13 (AY-3-8910)
non disponibili sull'AY-3-8912

Ingresso/uscita A7--A0, B7--B0
Ciascuna di queste due porte d'ingresso/uscita parallele consentono lo scambio di dati di 8 bit in parallelo da/per il bus CPU/PSG da/a un qualsiasi dispositivo esterno connesso ai piedini IOA o IOB. Ciascun piedino è fornito di un resistore "pull-up", in modo che quando si trovano nella modalità d'ingresso tutti i piedini hanno normalmente il segnale alto. Il metodo raccomandato quindi per la scansione di selettori esterni è quello di mettere a terra il bit d'ingresso.

TEST 1: piedino 39 (AY-3-8910)
piedino 2 (AY-3-8912)
TEST 2: piedino 26 (AY-3-8910)
non connesso nell'AY-3-8912;
Questi piedini sono usati a scopi di controllo dalla General Instruments e devono quindi essere lasciati aperti.
V (CC): piedino 40 (AY-3-8910)
piedino 3 (AY-3-8912)
Alimentazione nominale +5 Volt del PSG.
V (SS): piedino 1 (AY-3-8910)
piedino 6 (AY-3-8912)
Riferimento di terra per il PSG.

## 2.4 Temporizzazione del bus

Poiché le funzioni del PSG sono controllate per mezzo di comandi dal processore di sistema, il bus comune tra dati ed indirizzi (DA7--DA0) richiede di essere definito in dipendenza dalla funzione attiva in un dato momento. Questa condizione viene soddisfatta attraverso i segnali di controllo stanziati dal processore, descritti in precedenza, che definiscono lo stato del bus; il PSG decodifica quindi questi segnali per effettuare la funzione richiesta.
L'impostazione di questi segnali di controllo da parte del processore è la medesima del caso in cui il processore interagisce con una RAM: (1) il processore imposta l'indirizzo di memoria; (2) il processore scrive o legge il dato sulla/dalla memoria. Nel nostro caso la memoria è il vettore dei 16 registri lettura/scrittura di controllo del PSG.
Le relazioni temporali nelle istanze dei segnali di controllo del bus vengono esaminate in generale dalla seguente sezione, e in dettaglio nella sezione 7, "Specifiche elettriche".

*Fig. 3 - Vettore registri del PSG.*

| REGISTRO | BIT | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| R0 | Periodo suono canale A | 8 bit regolazione accurata A | | | | | | | |
| R1 | | | | | | 4 bit regolaz. di mass. A | | | |
| R2 | Periodo suono canale B | 8 bit regolazione accurata B | | | | | | | |
| R3 | | | | | | 4 bit regolaz. di mass. B | | | |
| R4 | Periodo suono canale C | 8 bit regolazione accurata C | | | | | | | |
| R5 | | | | | | 4 bit regolaz. di mass. C | | | |
| R6 | Periodo rumore | | | | 5 bit controllo del periodo | | | | |
| R7 | Abilitazione | IN/OUT | | Rumore | | | Suono | | |
| | | 10B | 10A | C | B | A | C | B | A |
| R10 | Ampiezza canale A | | | | M | L3 | L2 | L1 | L0 |
| R11 | Ampiezza canale B | | | | M | L3 | L2 | L1 | L0 |
| R12 | Ampiezza canale C | | | | M | L3 | L2 | L1 | L0 |
| R13 | Controllo periodo d'inviluppo | 8 bit regolazione accurata | | | | | | | |
| R14 | | 8 bit regolazione di massima | | | | | | | |
| R15 | Forma/ciclo d'inviluppo | | | | | CONT. | ATT. | ALT. | TEN. |
| R16 | Mem. dato porta A di I/O | I/O parallelo 8 bit sulla porta A | | | | | | | |
| R17 | Mem. dato porta B di I/O | I/O parallelo 8 bit sulla porta B | | | | | | | |

| NOTA | OTTAVA | FREQUENZA IDEALE | FREQUENZA REALE | VALORE OTTALE 12 BIT DEL REGISTRO | | | | NOTA | OTTAVA | FREQUENZA IDEALE | FREQUENZA REALE | VALORE OTTALE 12 BIT DEL REGISTRO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | 1 | 32.703 | 32.698 | 6 | 5 | 3 | 5 | C | 6 | 523.248 | 522.714 | 0 | 3 | 2 | 6 |
| C# | 1 | 34.648 | 34.653 | 6 | 2 | 3 | 4 | C# | 5 | 554.368 | 553.766 | 0 | 3 | 1 | 2 |
| D | 1 | 36 708 | 36.712 | 5 | 7 | 4 | 7 | D | 5 | 587.328 | 588.741 | 0 | 2 | 7 | 6 |
| D# | 1 | 38.891 | 38.895 | 5 | 4 | 7 | 4 | D# | 5 | 622.256 | 621.449 | 0 | 2 | 6 | 4 |
| E | 1 | 41.203 | 41.201 | 5 | 2 | 3 | 3 | E | 5 | 659.248 | 658.005 | 0 | 2 | 5 | 2 |
| F | 1 | 43.654 | 43.662 | 5 | 0 | 0 | 2 | F | 5 | 698.464 | 699.130 | 0 | 2 | 4 | 0 |
| F# | 1 | 46.249 | 46.243 | 4 | 5 | 6 | 3 | F# | 5 | 739.984 | 740.800 | 0 | 2 | 2 | 7 |
| G | 1 | 48.999 | 48 997 | 4 | 3 | 5 | 3 | G | 5 | 783.984 | 782.243 | 0 | 2 | 1 | 7 |
| G# | 1 | 51.913 | 51 908 | 4 | [illegible] | 5 | 3 | G# | 5 | 830.608 | 828 598 | 0 | 2 | 0 | 7 |
| A | 1 | 55.000 | 54.995 | 3 | 7 | 6 | 2 | A | 5 | 880.000 | 880.794 | 0 | 1 | 7 | 7 |
| A# | 1 | 58.270 | 58.261 | 3 | 6 | 0 | 0 | A# | 5 | 932.320 | 932.173 | 0 | 1 | 7 | 0 |
| B | 1 | 61.735 | 61.733 | 3 | 4 | 2 | 4 | B | 5 | 987.760 | 989.918 | 0 | 1 | 6 | 1 |
| C | 2 | 65 406 | 65.416 | 3 | 2 | 5 | 6 | C | 6 | 1046.496 | 1045.428 | 0 | 1 | 5 | 3 |
| C# | 2 | 69 296 | 69.307 | 3 | 1 | 1 | 6 | C# | 6 | 1108 736 | 1107.532 | 0 | 1 | 4 | 5 |
| D | 2 | 73.416 | 73.399 | 2 | 7 | 6 | 4 | D | 6 | 1174.656 | 1177.482 | 0 | 1 | 3 | 7 |
| D# | 2 | 77 782 | 77.789 | 2 | 6 | 3 | 6 | D# | 6 | 1244.512 | 1242.898 | 0 | 1 | 3 | 2 |
| E | 2 | 82.406 | 82 432 | 2 | 5 | 1 | 5 | E | 6 | 1318.496 | 1316.009 | 0 | 1 | 2 | 5 |
| F | 2 | 87.308 | 87.323 | 2 | 4 | 0 | 1 | F | 6 | 1396.928 | 1398.260 | 0 | 1 | 2 | 0 |
| F# | 2 | 92 498 | 92.523 | 2 | 2 | 7 | 1 | F# | 6 | 1479.968 | 1471.852 | 0 | 1 | 1 | 4 |
| G | 2 | 97 998 | 98.037 | 2 | 1 | 6 | 5 | G | 6 | 1567.968 | 1575.504 | 0 | 1 | 0 | 7 |
| G# | 2 | 103 826 | 103 863 | 2 | 0 | 6 | 5 | G# | 6 | 1661.216 | 1669.564 | 0 | 1 | 0 | 3 |
| A | 2 | 110.000 | 109.991 | 1 | 7 | 7 | 1 | A | 6 | 1760.000 | 1747.825 | 0 | 1 | 0 | 0 |
| A# | 2 | 116.540 | 116 522 | 1 | 7 | 0 | 0 | A# | 6 | 1864.640 | 1864.346 | 0 | 0 | 7 | 4 |
| B | 2 | 123 470 | 123.467 | 1 | 6 | 1 | 2 | B | 6 | 1975.520 | 1962 470 | 0 | 0 | 7 | 1 |
| C | 3 | 130 812 | 130 831 | 1 | 5 | 2 | 7 | C | 7 | 2092.992 | 2110.581 | 0 | 0 | 6 | 5 |
| C# | 3 | 138.592 | 138 613 | 1 | 4 | 4 | 7 | C# | 7 | 2217.472 | 2237.216 | 0 | 0 | 6 | 2 |
| D | 3 | 146 832 | 146 799 | 1 | 3 | 7 | 2 | D | 7 | 2349.312 | 2330.433 | 0 | 0 | 6 | 0 |
| D# | 3 | 155 564 | 155.578 | 1 | 3 | 1 | 7 | D# | 7 | 2489.024 | 2485.795 | 0 | 0 | 5 | 5 |
| E | 3 | 164 812 | 164 743 | 1 | 2 | 4 | 7 | E | 7 | 2636.992 | 2663.352 | 0 | 0 | 5 | 2 |
| F | 3 | 174.616 | 174.510 | 1 | 2 | 0 | 1 | F | 7 | 2793.856 | 2796.520 | 0 | 0 | 5 | 0 |
| F# | 3 | 184.996 | 184.894 | 1 | 1 | 3 | 5 | F# | 7 | 2959.936 | 2943.705 | 0 | 0 | 4 | 6 |
| G | 3 | 195 996 | 195.903 | 1 | 0 | 7 | 3 | G | 7 | 3135.936 | 3107.244 | 0 | 0 | 4 | 4 |
| G# | 3 | 207 652 | 207 534 | 1 | 0 | 3 | 3 | G# | 7 | 3322.432 | 3290.023 | 0 | 0 | 4 | 2 |
| A | 3 | 220.000 | 220 198 | 0 | 7 | 7 | 4 | A | 7 | 3520.000 | 3495.649 | 0 | 0 | 4 | 0 |
| A# | 3 | 233 080 | 233 043 | 0 | 7 | 4 | 0 | A# | 7 | 3729.280 | 3728.693 | 0 | 0 | 3 | 6 |
| B | 3 | 246.940 | 246 933 | 0 | 7 | 0 | 5 | B | 7 | 3951.040 | 3995 028 | 0 | 0 | 3 | 4 |
| C | 4 | 261.624 | 261.357 | 0 | 6 | 5 | 4 | C | 8 | 4185.984 | 4142.992 | 0 | 0 | 3 | 3 |
| C# | 4 | 277 184 | 276.883 | 0 | 6 | 2 | 4 | C# | 8 | 4434.944 | 4474.431 | 0 | 0 | 3 | 1 |
| D | 4 | 293.664 | 293.598 | 0 | 5 | 7 | 5 | D | 8 | 4698.624 | 4660.866 | 0 | 0 | 3 | 0 |
| D# | 4 | 311.128 | 310.724 | 0 | 5 | 5 | 0 | D# | 8 | 4978 048 | 5084.581 | 0 | 0 | 2 | 6 |
| E | 4 | 329.624 | 329 973 | 0 | 5 | 2 | 3 | E | 8 | 5273.984 | 5326.704 | 0 | 0 | 2 | 5 |
| F | 4 | 349.232 | 349 565 | 0 | 5 | 0 | 0 | F | 8 | 5587.712 | 5593.039 | 0 | 0 | 2 | 4 |
| F# | 4 | 369.992 | 370.400 | 0 | 4 | 5 | 6 | F# | 8 | 5919.872 | 5887.410 | 0 | 0 | 2 | 3 |
| G | 4 | 391.992 | 392 494 | 0 | 4 | 3 | 5 | G | 8 | 6271.872 | 6214.488 | 0 | 0 | 2 | 2 |
| G# | 4 | 415 304 | 415 839 | 0 | 4 | 1 | 5 | G# | 8 | 6644 864 | 6580.046 | 0 | 0 | 2 | 1 |
| A | 4 | 440.000 | 440 397 | 0 | 3 | 7 | 6 | A | 8 | 7040 000 | 6991.299 | 0 | 0 | 2 | 0 |
| A# | 4 | 466 160 | 466 087 | 0 | 3 | 6 | 0 | A# | 8 | 7458.560 | 7457.365 | 0 | 0 | 1 | 7 |
| B | 4 | 493.880 | 494.959 | 0 | 3 | 4 | 2 | B | 8 | 7902.080 | 7990.056 | 0 | 0 | 1 | 6 |

*Fig. 23 - Scala cromatica di temperamento equafile (frequenza temporizzazione = 1.78977 MHz).*

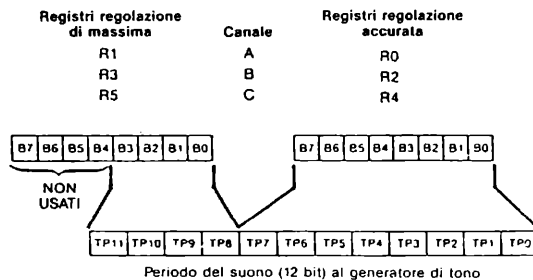
## 3. Guida operativa

Poiché tutte le funzioni del PSG sono controllate dal processore ospite per mezzo di una serie di caricamenti dei registri, una descrizione dettagliata delle operazioni del PSG può ottenersi ponendo in relazione ciascuna funzione con il valore di controllo impostato nell'opportuno registro. La creazione di un programma per ottenere un particolare suono o effetto sonoro segue la sequenza di controllo che viene riportata di seguito:

| *Sezione* | *Operazione* | *Registri* | *Funzione* |
|---|---|---|---|
| 3.1 | Controllo del generatore di suono | R0-R5 | Programmazione dei periodi del suono |
| 3.2 | Controllo del generatore di rumore | R6 | Programmazione del periodo del rumore |
| 3.3 | Controllo del mixer | R7 | Abilita suono e/o rumore sui canali selezionati |
| 3.4 | Controllo d'ampiezza | R10-R12 | Seleziona ampiezze fisse o a inviluppo variabile |
| 3.5 | Controllo generatore d'inviluppo | R13-R15 | Programmazione del periodo d'inviluppo e scelta della forma d'inviluppo |

### 3.1 Controllo generatore di suono (registri R0, R1, R2, R3, R4, R5)

La frequenza di ciascuna onda quadra prodotta dai tre generatori di suono (uno per ogni canale A, B, C) viene ottenuta nel PSG dapprima dividendo per 16 all'ingresso del CLOCK e poi dividendo ulteriormente il risultato per il valore del periodo del suono programmato su 12 bit. Ciascun valore su 12 bit viene ottenuto nel PSG combinando il contenuto dei registri di regolazione di massima e di regolazione accurata, come illustrato nella figura che segue:

Periodo del suono (12 bit) al generatore di tono

Si noti che il valore su 12 bit ottenuto dalla combinazione dei due registri di regolazione indica un PERIODO - più alto è questo valore e minore sarà la frequenza.
Si noti inoltre che a causa della tecnica adottata per il calcolo del periodo del suono, il valore più basso è 000000000001 (divide per 1) mentre quello più alto è 111111111111 (divide per 4095).
Le equazioni che descrivono le relazioni tra la frequenza desiderata del suono in uscita e la frequenza d'ingresso del CLOCK e del periodo del suono sono le seguenti:

(a) $f_T = \dfrac{f_{CLOCK}}{16TP_{10}}$ (b) $TP_{10} = 256CT_{10} + FT_{10}$

dove:
$f_T$ = frequenza di suono desiderata
$f_{CLOCK}$ = frequenza del CLOCK in ingresso
$TP_{10}$ = decimale equivalente ai bit del periodo del suono TP11--TP0
$CT_{10}$ = decimale equivalente ai bit B3--B0 del registro di regolazione di massima (TP11--TP8)
$FT_{10}$ = decimale equivalente ai bit B7--B0 del registro di regolazione accurata (TP7--TP0).
Dalle equazioni di cui sopra si può dedurre che la frequenza di suono può variare da un estremo inferiore di F(CLOCK)/65520 (con $TP_{10}=4095_{10}$) ad un estremo superiore F(CLOCK)/16 (con $TP_{10}=1$). Se si usa un temporizzatore in ingresso che funziona a 2MHz, per esempio, si può ottenere un intervallo di frequenza tra 30,5 Hz a 125KHz.
Per calcolare i valori da impostare nei registri di regolazione di massima del periodo del suono e di regolazione accurata dello stesso, conoscendo l'ingresso del CLOCK e la frequenza di uscita desiderata, è sufficiente manipolare le equazioni viste prima, ottenendo:

(a) $TP_{10} = \dfrac{f_{CLOCK}}{16\ f_T}$ (b) $CT_{10} + \dfrac{FT_{10}}{256} = \dfrac{TP_{10}}{256}$

Esempio 1
$f_T$ = 1KHz
$f_{CLOCK}$ = 2MHz

$$TP_{10} = \frac{2 \times 10^6}{16(1 \times 10^3)} = 125$$

Sostituendo nell'equazione (b):

$$CT_{10} + \frac{FT_{10}}{256} = \frac{125}{256}$$

da cui:

$CT_{10} = 0 \quad = 0000$ (B3--B0)
$FT_{10} = 125_{10} = 01111101$ (B7--B0)

Esempio 2

$f_T = 100$ Hz
$f_{CLOCK} = 2$MHz

$$TP_{10} = \frac{2\times10^6}{16(1\times10^2)} = 1250$$

Sostituendo nell'equazione (b):

$$CT_{10} + \frac{FT_{10}}{256} = \frac{1250}{256} = 4 + \frac{226}{256}$$

da cui:

$CT_{10} = 4_{10} \quad = 0100$ (B3--B0)
$FT_{10} = 226_{10} = 11100010$ (B7--B0)

### 3.2 Controllo del generatore di rumore (registro R6)

La frequenza della sorgente di rumore viene ottenuta nel PSG dapprima dividendo per 16 il segnale di CLOCK in ingresso, poi dividendo ulteriormente il risultato per il valore programmato su 5 bit del periodo del rumore. Questo valore è contenuto nei 5 bit meno significativi del registro R6 (B4--B0), come viene illustrato di seguito

Periodo del rumore,
REGISTRO 6

| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|
| NON USATI | | | NP (Periodo rumore) al generatore di rumore | | | | |

Si noti che il valore su 5 bit è un periodo: quindi più alto è questo valore e minore sarà la frequenza di rumore risultante. Si noti anche che, come per il periodo del suono, il valore più basso è 00001 (divide per 1) e quello più alto è 11111 (divide per 31).
L'equazione per la frequenza del rumore è la seguente:

$$f_N = \frac{f_{CLOCK}}{16\ NP_{10}}$$

Dove:
$f_N$ = frequenza di rumore desiderata
$f_{CLOCK}$ = frequenza del CLOCK in ingresso
$NP_{10}$ = decimale equivalente ai bit B4--B0 del registro del periodo del rumore.
Dalla precedente equazione si può dedurre che la frequenza del rumore può variare da un estremo inferiore di $f_{CLOCK}/496$ (per $NP_{10} = 31_{10}$) a un estremo superiore di $f_{CLOCK}/16$ (per $NP_{10} = 1$). Se si usa un ingresso di CLOCK di 2MHz, per esempio, è possibile produrre una frequenza di rumore compresa tra 4KHz e 125 KHz.
Per calcolare il valore da impostare nel registro di periodo del rumore, conoscendo l'ingresso del CLOCK e la frequenza desiderata in uscita per il rumore, basta manipolare l'equazione di cui sopra nel seguente modo:

$$NP_{10} = \frac{f_{CLOCK}}{16\ f_N}$$

### 3.3 Controllo del mixer - Abilitazione dell'I/O (registro 7)

Il registro 7 è un registro di abilitazione multifunzione che controlla i tre mixer suono/rumore e le due porte di I/O di uso generale.
I mixer, come accennato in precedenza, combinano le frequenze del tono e del rumore per ciascuno dei tre canali. La decisione di combinare tono e rumore o nessuno dei due per ciascun canale viene memorizzata dai bit B5--B0 del registro R7.
La direzione (ingresso o uscita) delle due porte di I/O ad uso generale (IOA

e IOB) viene memorizzata dallo stato dei bit B7 e B6 del medesimo registro. Queste funzioni vengono illustrate qui di seguito:

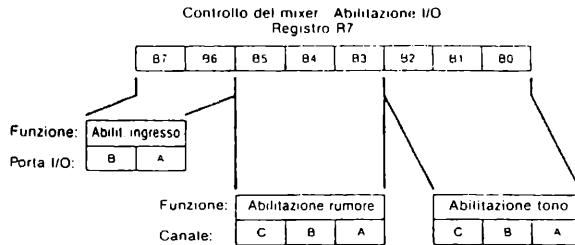



*Tavola di verità per l'abilitazione del rumore*

| *Bit di R7* | | | *Rumore abilitato sul canale* | | |
|---|---|---|---|---|---|
| *B5* | *B4* | *B3* | | | |
| 0 | 0 | 0 | C | B | A |
| 0 | 0 | 1 | C | B | — |
| 0 | 1 | 0 | C | — | A |
| 0 | 1 | 1 | C | — | — |
| 1 | 0 | 0 | — | B | A |
| 1 | 0 | 1 | — | B | — |
| 1 | 1 | 0 | — | — | A |
| 1 | 1 | 1 | — | — | — |

*Tavola di verità per l'abilitazione del tono*

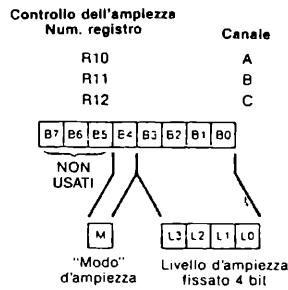
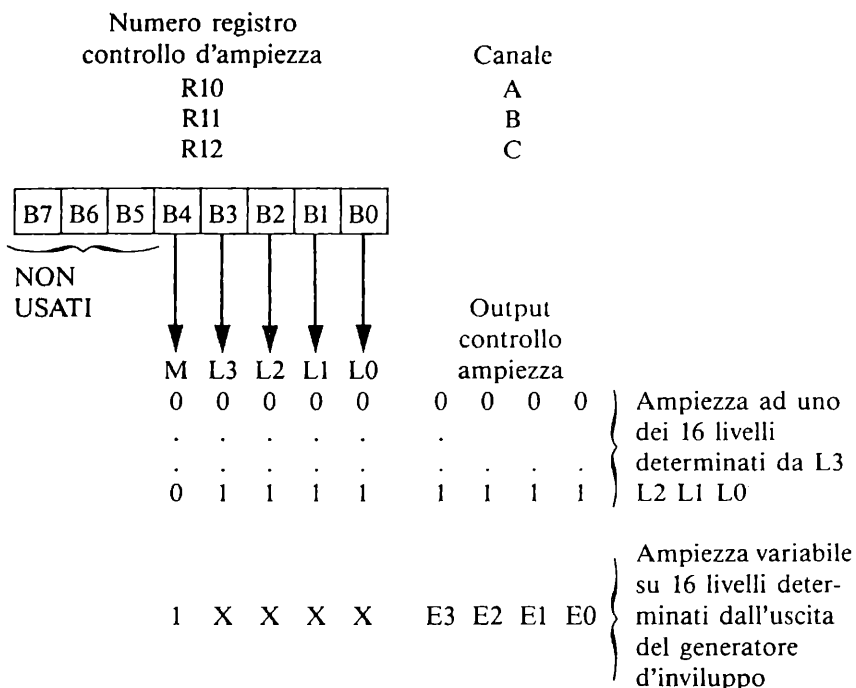
| *Bit di R7* | | | *Tono abilitato sul canale* | | |
|---|---|---|---|---|---|
| *B2* | *B1* | *B0* | | | |
| 0 | 0 | 0 | C | B | A |
| 0 | 0 | 1 | C | B | — |
| 0 | 1 | 0 | C | — | A |
| 0 | 1 | 1 | C | — | — |
| 1 | 0 | 0 | — | B | A |
| 1 | 0 | 1 | — | B | — |
| 1 | 1 | 0 | — | — | A |
| 1 | 1 | 1 | — | — | — |

*Tavola di verità per la porta di I/O*

| *Bit di R7* | | *Stato porta I/O* | |
|---|---|---|---|
| *B7* | *B6* | *IOB* | *IOA* |
| 0 | 0 | Input | Input |
| 0 | 1 | Input | Output |
| 1 | 0 | Output | Input |
| 1 | 1 | Output | Output |

Nota: La disabilitazione del suono e del rumore non "spegne" un canale. Questa operazione può essere compiuta solo scrivendo tutti 0 nel corrispondente registro di controllo d'ampiezza (R10, R11 e R12; si veda 3.4).

### 3.4 Controllo d'ampiezza (registro R10, R11, R12)

L'ampiezza dei segnali generati da ciascuno dei tre convertitori D/A (uno per ogni canale, A, B e C) è determinata da cinque bit meno significativi (B4--B0) dei registri R10, R11 e R12 come illustrato:

La "modalità" d'ampiezza (bit M) determina se l'ampiezza deve essere fissa (M=0) oppure se il suo livello deve essere variabile (M=1). Ovviamente da ciò consegue che i bit L3--L0 che definiscono il valore di un'ampiezza fissa hanno significato solo nel caso in cui M=0. In questo caso il livello dell'ampiezza è "fissato" solo nel senso che esso è sotto controllo diretto del processore di sistema (attraverso i bit D3--D0). La modifica dell'ampiezza quando si è nella modalità "fissa" comporta in ogni istante l'intervento del processore di sistema attraverso una sequenza indirizzo di allacciamento/dato da scrivere per modificare il contenuto di D3--D0.
Quando M=1 (livello d'ampiezza variabile), l'ampiezza di ciascun canale è determinata dal modello d'inviluppo così come è definito dai quattro bit di output E3, E2, E1 e E0 del generatore d'inviluppo.
La "modalità" d'ampiezza (bit M) può essere anche pensata come un bit di "abilitazione dell'inviluppo"; cioè quando M=0 esso non viene usato, e viceversa è abilitato quando M=1. Una descrizione completa della funzione del generatore d'inviluppo segue nella sezione 3.5.
Il diagramma completo che descrive tutte le combinazioni dei 5 bit del controllo d'ampiezza è mostrato di seguito:

La Figura 6 illustra graficamente la selezione di un'ampiezza a livello variabile (controllata dall'inviluppo) dove i 16 livelli sono in diretta corrispondenza con l'uscita del generatore d'inviluppo. Un'ampiezza a livello "fissato" corrisponderebbe a uno solo di questi livelli, quello determinato dal decimale equivalente alla configurazione dei bit L3, L2, L1 e L0.

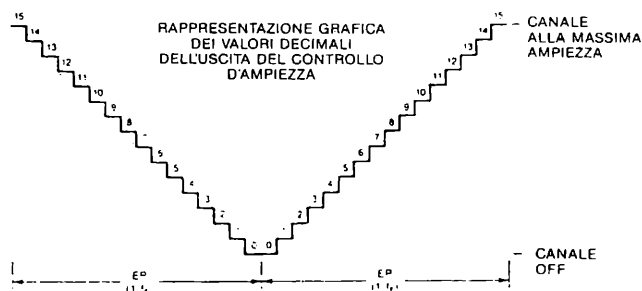
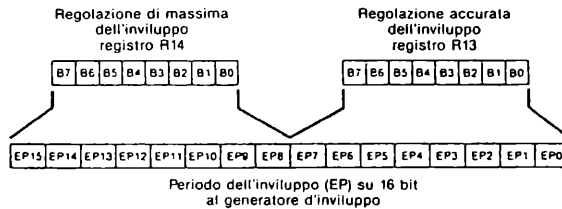



*Fig. 6 - Controllo d'ampiezza variabile (M = 1).*

## 3.5 Controllo del generatore d'inviluppo (registri R13, R14, R15)

Per ottenere la generazione di modelli d'inviluppo piuttosto complessi, nel PSG sono disponibili due metodi indipendenti di controllo. Con il primo è possibile variare la frequenza dell'inviluppo usando i registri R13 e R14; con il secondo la forma relativa e la forma del ciclo dell'inviluppo possono essere variate usando il registro R15. Nei seguenti paragrafi verranno illustate nei dettagli le funzioni di controllo dell'inviluppo: dapprima si parlerà del controllo del periodo e poi del controllo della forma/ciclo.

### 3.5.1 Controllo del periodo dell'inviluppo (registri R13, R14)

La frequenza dell'inviluppo viene calcolata, nel PSG, dapprima dividendo per 256 l'ingresso di CLOCK, quindi dividendo ulteriormente il risultato dal valore a 16 bit programmato come periodo d'inviluppo. Nel PSG questo valore su 16 bit viene ottenuto combinando i contenuti dei registri di regolazione accurata e di massima dell'inviluppo, come illustrato di seguito:

Si noti che il valore programmato su 16 bit nei registri suddetti è un periodo: più alto è questo valore e minore sarà la frequenza d'inviluppo che si ottiene.
Si noti inoltre che, come per il periodo del Tono, il valore minimo è 0000000000000001 (divide per 1) e quello massimo è 1111111111111111 (divide per $65535_{10}$).
Le equazioni che regolano la frequenza d'inviluppo sono:

(a) $f_E = \dfrac{f_{CLOCK}}{256EP_{10}}$ (b) $EP_{10} = 256CT_{10} + FT_{10}$

Dove:
$f_E$ è la frequenza d'inviluppo voluta
$f_{CLOCK}$ è la frequenza dell'ingresso di CLOCK

$EP_{10}$ è il decimale equivalente ai bit del periodo d'inviluppo EP15--EP0
$CT_{10}$ è il decimale equivalente ai bit B7--B0 (EP15 -- EP8) del registro di regolazione di massima
$FT_{10}$ è il decimale equivalente ai bit B7--B0 (EP7--EP0) del registro di regolazione accurata.
Da queste equazioni si può dedurre che la frequenza d'inviluppo varia tra un estremo inferiore di $f_{CLOCK}/16776960$ (dove $EP_{10} = 65535_{10}$) a un estremo superiore di $f_{CLOCK}/256$ (dove $EP_{10} = 1$). Usando per esempio un clock a 2MHz si ottiene un intervallo per la frequenza d'inviluppo da 0,12 Hz a 7812,5 Hz.
Per calcolare i valori contenuti nei registri di regolazione di massima e accurata del periodo d'inviluppo, conoscendo il CLOCK d'ingresso e la frequenza che si desidera ottenere, è sufficiente manipolare come segue le equazioni viste prima:

(a) $$EP_{10} = \frac{f_{CLOCK}}{256 f_E}$$ (b) $$CT_{10} + \frac{FT_{10}}{256} = \frac{EP_{10}}{256}$$

Esempio:
$f_E = 0{,}5$ Hz
$f_{CLOCK} = 2$ MHz

$$EP_{10} = \frac{2 \times 10^6}{256(0{,}5)} = 15.625$$

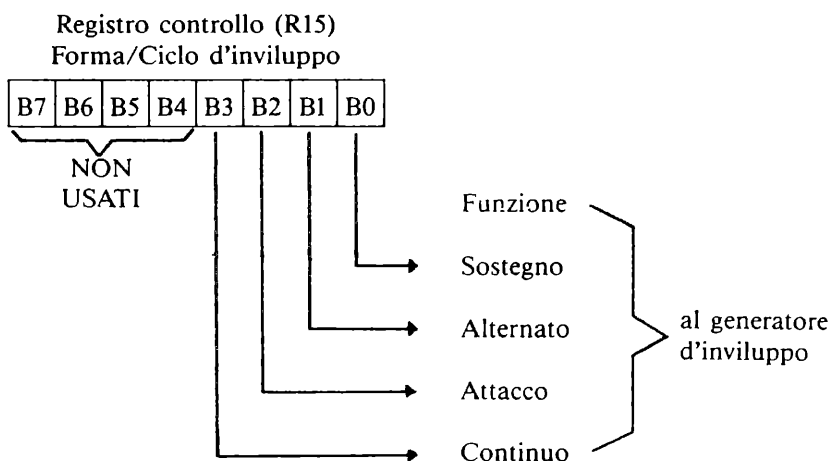
Sostituendo nell'equazione (b):

$$CT_{10} + \frac{FT_{10}}{256} = \frac{15.625}{256} = 61 + \frac{9}{256}$$

da cui si ricava:
$CT_{10} = 61_{10} = 00111101$ (B7--B0)
$FT_{10} = 9_{10} = 00001001$ (B7--B0)

### 3.5.2 Controllo forma/ciclo dell'inviluppo (registro R15)

Il generatore d'inviluppo divide inoltre la frequenza d'inviluppo per 16 producendo un modello d'inviluppo a 16 stati per ciclo così com'è definito dal suo contatore su 4 bit, E3 E2 E1 e E0. La particolare forma e modello di ciclo di un qualsiasi inviluppo voluto viene ottenuta controllando il modello di conteggio (incremento o decremento) del contatore su 4 bit e definendo un modello a singolo-ciclo o a ciclo-ripetuto.

Questo controllo forma/ciclo viene memorizzato nei 4 bit meno significativi (B3 -- B0) del registro R15. Ciascuno di questi bit controlla una funzione del generatore d'inviluppo, nel modo di seguito illustrato:

La definizione di ogni funzione è la seguente:
Sostegno - Quando vale 1, limita l'inviluppo a un solo ciclo mantenendo l'ultimo valore del contatore d'inviluppo (E3--E0 = 0000 o 1111 secondo la modalità del contatore (incremento o decremento, rispettivamente).
Alternato - Quando vale 1 il contatore d'inviluppo inverte la modalità di conteggio dopo ogni ciclo.
Nota: quando sono a 1 sia il bit Sostegno che quello Alternato, il contatore d'inviluppo viene rinizializzato prima della memorizzazione dell'ultimo valore.
Attacco - Quando vale 1 il contatore d'inviluppo si incrementerà (attacco) da E3 E2 E1 E0 = 0000 a E3 E2 E1 E0 = 1111. Viceversa quando vale zero il conteggio andrà da 1111 a 0000.
Continuo - Quando vale 1 il modello del ciclo è definito dal bit di Sostegno. Quando vale 0 il generatore d'inviluppo si rinizializzerà a 0000 dopo ogni ciclo e manterrà quel valore.
Un'ulteriore descrizione delle funzioni accennate potrebbe essere ottenuta con numerosi diagrammi in cui illustrare le sequenze di conteggio binario per ogni combinazione di Sostegno, Alternato, Attacco e Continuo. Poiché però queste uscite vengono usate (quando selezionate dai registri di controllo d'ampiezza) per modulare in ampiezza le uscite dai mixers, una miglior comprensione dei loro effetti può ottenersi attraverso una rappresen-

tazione grafica dei loro valori per ogni condizione selezionata, come illustrato nelle figura 7 e 8.

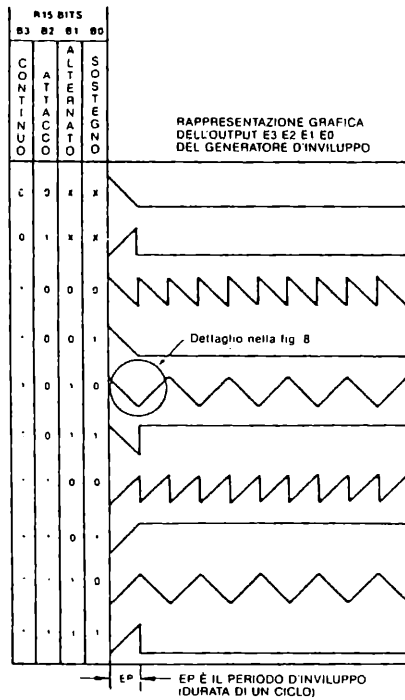



*Fig. 7 - Controllo forma/ciclo dell'inviluppo.*

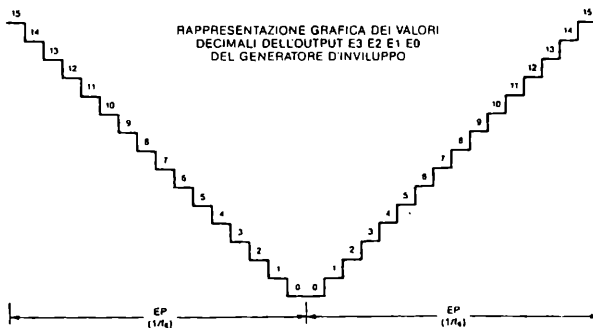



*Fig. 8 - Dettaglio di due cicli della figura 7 (forma d'onda "1010").*

## 6 Generazione di suoni musicali

Il PSG viene usato principalmente per produrre suoni non musicali come effetti che accompagnino l'azione visiva. Nelle seguenti sezioni vengono descritte delle tecniche e vengono forniti esempi per la creazione degli effetti più famosi ed utilizzati. Tutti gli esempi fanno riferimento a un temporizzatore funzionante a 1,78977 MHz.

### 6.1 Effetti di solo suono

Molti effetti sono producibili usando solamente le capacità di generazione di suono del PSG, senza cioè il rumore o la generazione d'inviluppo. Esempi di effetti di questo genere sono le frequenze di un suono di segnale telefonico (due frequenze distinte prodotte simultaneamente) o l'effetto della sirena europea (due frequenze distinte prodotte sequenzialmente): segue la lista dei valori dei registri di controllo atti all'ottenimento degli effetti citati.

*Fig. 27 - Diagramma per l'effetto "Sirena europea"*

| Numero registro | Valore ottale da caricare | Spiegazione |
|---|---|---|
| Tutti i non specif. | 000 | — |
| R0 | 376 } | Periodo suono canale A = 2.27 ms |
| R1 | 000 } | (440 Hz) |
| R7 | 076 | Abilita solo suono canale A |
| R10 | 017 | Seleziona massima ampiezza canale A |
| | (attendere circa 350 ms prima di continuare) | |
| R0 | 126 } | Periodo suono canale A = 5,346 ms |
| R1 | 001 } | (187Hz) |
| | (attendere circa 350 ms prima di continuare) | |
| R10 | 000 | Spegne canale A; termina l'effetto |

### 6.2 Effetti di solo rumore

Alcuni degli effetti sonori più comunemente richiesti implicano l'uso del generatore di rumore e del generatore d'inviluppo (oppure il controllo da parte del processore dell'inviluppo di canale se altri canali stanno usando il generatore d'inviluppo).

Esempi di quanto detto sono gli effetti "sparo" ed "esplosione", che vengono illustrati nelle figure 28 e 29. In entrambi i casi viene usato solo il rumore con un inviluppo di decadimento. Negli esempi riportati le uniche differenze sono la lunghezza dell'inviluppo, modificata attraverso il registro di regolazione di massima, e il periodo del rumore. Si noti che una "esplosione" significativamente minore può essere ottenuta con tutti e tre i canali che operano con gli stessi parametri.

*Fig. 28 - Diagramma per l'effetto "sparo"*

| Numero registro | Valore ottale da caricare | Spiegazione |
|---|---|---|
| Tutti i non specif. | 000 | — |
| R6 | 017 | Valore medio per periodo rumore |
| R7 | 007 | Abilita solo rumore su A, B, C |
| R10 | 020 | Seleziona intervallo completo |
| R11 | 020 | d'ampiezza sotto controllo diretto |
| R12 | 020 | del generatore inviluppo |
| R14 | 020 | Periodo inviluppo 0,586 secondi |
| R15 | 000 | Seleziona inviluppo di decadimento un solo ciclo |

*Fig. 29 - Diagramma dell'effetto "esplosione"*

| Numero registro | Valore ottale da caricare | Spiegazione |
|---|---|---|
| Tutti i non specif. | 000 | — |
| R6 | 000 | Imposta il periodo del rumore al massimo valore |
| R7 | 007 | Abilita solo il rumore canali A, B, C |
| R10 | 020 | Seleziona intervallo completo |
| R11 | 020 | d'ampiezza sotto controllo diretto |
| R12 | 020 | del generatore inviluppo |
| R14 | 070 | Imposta periodo inviluppo a 2,05 sec. |
| R15 | 000 | Seleziona inviluppo di decadimento un solo ciclo |

## 6.3 Effetti con variazione continua di frequenza

Gli effetti sonori "laser", "sibilo di bomba", "ululato del lupo" e "auto da corsa" (illustrati nelle figure dalla 30 alla 33) utilizzano effetti di variazio-

ne continua della frequenza. In tutti i casi si opera un incremento o un decremento dei valori contenuti nei registri di periodo del suono in base a un inizio, una fine e un tempo tra i cambiamenti di frequenza, di volta in volta diversi. Per esempio la variazione dell'effetto laser è molto più rapida dell'effetto accelerazione dell'auto da corsa: ciò nonostante usano, con differenti parametri, la stessa routine.
Altri risultati facilmente ottenibili sono delle variazioni continue di rumore o l'effetto doppler. La variazione continua del registro di temporizzazione del rumore (R6) produce un effetto doppler che sembra particolarmente adatto ai giochi di tipo "guerre stellari".

*Fig. 30 - Diagramma effetto sonoro "laser"*

| **Numero registro** | **Valore ottale da caricare** | **Spiegazione** |
|---|---|---|
| Tutti i non specif. | 000 | — |
| R7 | 076 | Abilita il suono solo sul canale A |
| R10 | 017 | Massima ampiezza canale A. |
| | | Effetto variazione continua (sweep) sul periodo suono canale A con ciclo |
| R0 | 060 (inizio) | processore, con tempo attesa ca. 3 ms |
| R0 | 160 (fine) | tra ogni passo da 060 a 160 |
| | | (da 0,429 ms/2330 Hz a 1 ms/1000 Hz) |
| R10 | 000 | Spegne canale A: l'effetto termina |

*Fig. 31 - Diagramma effetto sonoro "sibilo di una bomba"*

| **Numero registro** | **Valore ottale da caricare** | **Spiegazione** |
|---|---|---|
| Tutti i non specif. | 000 | — |
| R7 | 076 | Abilita il suono solo sul canale A |
| R10 | 017 | Massima ampiezza canale A. |
| | | Effetto variazione continua (sweep) sul periodo suono canale A con ciclo |
| R0 | 060 (inizio) | processore, con tempo attesa ca. 25 ms |
| R0 | 300 (fine) | tra ogni passo da 060 a 300 |
| | | (da 0,429 ms/2330 Hz a 1,72 ms/582 Hz) |
| R10 | 000 | Spegne Canale A: l'effetto termina |

Al termine del ciclo proseguire con la sequenza in figura 28

## 6.4 Effetti multicanale

Grazie all'indipendenza architetturale del PSG, molti effetti piuttosto complessi sono ottenibili senza sovraccaricare il processore. Per esempio, l'effetto "ululato del lupo" illustrato nella Figura 32 mostra l'uso di due canali per aggiungere un rumore costante di fiato ansimante alle tre variazioni continue di frequenza concentrate nell'ululato vero e proprio. Una volta che questo rumore è mandato sul canale, il processore deve solo effettuare la variazione continua della frequenza.

*Fig. 32 - Diagramma per l'effetto sonoro "ululato del lupo"*

| Numero registro | Valore ottale da caricare | Spiegazione |
|---|---|---|
| Tutti i non specif. | 000 | — |
| R6 | 001 | Valore minimo periodo rumore |
| R7 | 056 | suono canale A, rumore canale B |
| R10 | 017 | Massima ampiezza canale A |
| R11 | 011 | Ampiezza minore canale B. |
| R0<br>R0 | 100 (inizio)<br>040 (fine) | Effetto variazione continua sul periodo suono canale A con ciclo processore, con tempo attesa 12 ms ca. tra ogni passo da 100 a 040 (da 0,572 ms/1748 Hz a 0,286 ms/3496 Hz) |
| | (Attendere circa 150 ms prima di proseguire) | |
| R0<br>R0 | 100 (inizio)<br>060 (fine) | Ciclo del processore con tempo 25 ms ca. tra ogni passo da 100 a 060 ((da 0,572 ms/1748 Hz a 0,429 ms/233 Hz) |
| R0<br>R0 | 060 (inizio)<br>150 (fine) | Ciclo del processore con tempo ca. 6 ms tra ogni passo da 060 a 150 ((da 0,429 ms/233 Hz a 0,930 ms/1075 Hz) |
| R10<br>R11 | 000<br>000 | Termina l'effetto spegnendo canali A e B |

*Fig. 33 - Diagramma per l'effetto sonoro "auto da corsa"*

| **Numero registro** | **Valore ottale da caricare** | **Spiegazione** |
|---|---|---|
| Tutti i non specif. | 000 | — |
| R3 | 017 | Periodo suono canale B 34,33 ms/29 Hz |
| R7 | 074 | Abilita suono su canali A e B |
| R10 | 017 | Massima ampiezza canale A |
| R11 | 012 | Ampiezza minore canale B. |
| | | Effetto variazione continua |
| *R1/R0 | 013/000 (inizio) | con ciclo processore, con tempo attesa di |
| *R1/R0 | 004/000 (fine) | 3 ms ca. tra ogni passo da 013/000 a 004/000 |
| | | (da 25,17 ms/39,7 Hz a 9,15 ms/109,3 Hz). |
| | | Effetto variazione continua con ciclo |
| *R1/R0 | 011/000 (inizio) | processore, con tempo attesa ca. 3 ms |
| *R1/R0 | 003/000 (fine) | tra ogni passo da 011/000 a 003/000 |
| | | (da 20.6 ms/48.5 Hz a 6.97 ms/145.6 Hz). |
| | | Effetto variazione continua con ciclo |
| *R1/R0 | 006/000 (inizio) | processore con tempo attesa ca. 6 ms |
| *R1/R0 | 001/000 (fine) | tra ogni passo da 006/000 a 001/000 |
| | | (da 13,73 ms/72,8 Hz a 2,29 ms/436,7 Hz) |
| R10 | 000 | Termina l'effetto spegnendo canali A e B |
| R11 | 000 | |

* Decrementa R0/R1 come numero intero, cioè parte da 013/000, prosegue con 012/377, poi ancora 012/376 ecc.

Un compagno insostituibile per gli utilizzatori di computer MSX. Questo non è un libro per principianti, ma si rivolge a programmatori che operano sui computer MSX, spiegando esattamente come funzionano e come possono essere utilizzati al meglio.
La prima parte riguarda la progettazione di sistemi, la relazione tra linguaggio BASIC e MSX e l'introduzione al codice macchina.
Nella seconda parte viene illustrato l'impiego dell'Assembly nei computer MSX con particolare riguardo al processore Video, al Sound Chip di cui è dotata la macchina e alle funzioni di input/output.

L. [illegible] (IVA inclusa)

ISBN 88 7081 252 9

MSX linguaggio macchina

C.I. BURKINSHAW - R. GOODLEY